SHARP®

POCKET COMPUTER

MODEL

PC-1460

OPERATION MANUAL



Limits 69
Last Answer Feature 70
Length of Formula 72
Scientific Calculations in the BASIC mode 72
Direct Calculation Feature 76
Priority in Manual Calculation 78

CHAPTER 4 CONCEPTS AND TERMS OF BASIC 79
String Constants 79
Hexadecimal Numbers 79
Variables 80
Fixed Variables 81
Simple Variables 82
Array Variables 83
Variables in the Form of A ( ) 86
Expressions 87
Numeric Operators 88
String Expressions 88
Relational Expressions 88
Logical Expressions 89
Parentheses and Operator Precedence 91
RUN Mode 92
Functions 92

CHAPTER 5 PROGRAMMING THE COMPUTER 93
Programs 93
BASIC Statement 93
Line Numbers 93
BASIC Verbs 94
BASIC Commands 94
Modes 94
Beginning to Program on the COMPUTER 95
Example 1 – Entering and Running a Program 95
Example 2 – Editing a Program 96
Example 3 – Using Variables in programming 98
Example 4 – More Complex Programming 100
Storing Programs in the Memory 101

CHAPTER 6 SHORT CUTS 103
The DEF Key and Labelled Programs 103
Template 104

CHAPTER 7 USING PRINTERS AND CASSETTE INTERFACE 105
Using CE-126P Printer/Cassette interface 105
1. Using Printer 105

2. Using Cassette Interface 107
3. Cassette Tape Recorder 107
4. Operating the Cassette Interface and Recorder 109
Using Color Printer 113
Serial I/O Function 114

CHAPTER 8 USING THE RAM CARD 117
RAM Card Replacement 117
1. Replacing 1/2-Size RAM Card 118
2. Replacing Regular-Size RAM Card 120
Using RAM Card with Other Models 123

CHAPTER 9 BASIC REFERENCE 125
Commands 129
Verbs 146
Graphics Related Verbs 195
Functions 214
Pseudovariables 214
Numeric Functions 217
String Functions 223
Serial I/O Related Commands 225

CHAPTER 10 TROUBLESHOOTING 245
Machine Operation 245
BASIC Debugging 247

CHAPTER 11 MAINTENANCE OF THE COMPUTER 249

APPENDICES 251

APPENDIX A Error Messages 251

APPENDIX B Character Code Chart 255

APPENDIX C Formatting Output 257

APPENDIX D Expression Evaluation and Operator Priority 261

APPENDIX E Key Functions in Basic Mode 263

APPENDIX F Signals Used in the Serial I/O Terminal 269

APPENDIX G Specifications 271

APPENDIX H Using Programs Written on Other PC Models 273

PROGRAM EXAMPLES 279

INDEX 345

INTRODUCTORY NOTE

Welcome to the world of **SHARP** owners!

Few industries in the world today can match the rapid growth and technological advances being made in the field of personal computers. Computers which just a short time ago would have filled a huge room, required a Ph.D. to program, and cost thousands of dollars, now fit in the palm of your hand, are easily programmed, and cost so little that they are within the reach of nearly everyone.
Your new **SHARP** COMPUTER was designed to bring you all of the latest state-of-the-art features of this computing revolution and it incorporates many advanced capabilities:

* SCIENTIFIC CALCULATOR – It had been normal to use two different tasks, scientific calculation (including statistics) and computing, before this computer. But now only one tool is enough. The computer operates both as a scientific calculator and a pocket computer incorporating many programmed scientific functions plus BASIC command keys for simple programming.
* MEMORY SAFEGUARD – The computer remembers stored programs and variables even when you turn it off.
* Battery-powered operation for true portability.
* AUTO POWER OFF function which conserves the batteries by turning the power off if no activity takes place within a specified time limit.
* An expanded version of BASIC which provides formatted output, two dimensional arrays, variable length strings, and many other advanced features.
* Optional printers and cassette interfaces are available.

Congratulations on entering an exciting and enjoyable new world. We are sure that you will find this purchase one of the wisest you have ever made. The **SHARP COMPUTER** is a powerful tool, designed to meet your specific mathematical, scientific, engineering, business, and personal computing needs. With the **SHARP COMPUTER** you can begin NOW providing the solutions you'll need tomorrow!

Please note that the auxiliary RAM card is required for computer operation. If the power is turned on without the RAM card installed, none of the keys will function.

CHAPTER 1
HOW TO USE THIS MANUAL

This manual is designed to introduce you to the capabilities and features of your **COMPUTER** and to serve as a valuable reference tool. Whether you are a "first-time user" or an "old hand" with computers, you should acquaint yourself with the **COMPUTER** by reading and working through Chapters 2 through 6.

* Chapter 2 describes the physical features of the **COMPUTER**.
* Chapter 3 demonstrates the use of the **COMPUTER** as a scientific calculator.
* Chapter 4 defines some terms and concepts which are essential for BASIC programming, and tells you about the special considerations of these concepts on the **COMPUTER**.
* Chapter 5 introduces you to BASIC programming on the **COMPUTER**, showing you how to enter, correct, and run programs.
* Chapter 6 discusses some short cuts that make using your new **COMPUTER** easier and more enjoyable.

Chapter 9 is a reference section covering all the commands, verbs, and functions of BASIC that are grouped and alphabetically arranged within each group for your convenience.

Experienced BASIC programmers may go direct from Chapter 6 to Chapter 9 to learn the specific features of BASIC as implemented on the **COMPUTER**. Since every dialect of BASIC is somewhat different, read through this material at least once before starting serious programming.

If you have never programmed in BASIC before, we suggest that you buy a separate book on beginning BASIC programming or attend a BASIC class, before trying to work through these chapters. This manual is not intended to teach you how to program.

The remainder of the manual consists of:

* Chapter 7 — Basic information on the optional printers and cassette interfaces.
* Chapter 8 — An explanation on the use of the RAM card.
* Chapter 10 — A troubleshooting guide to help you solve some operating and programming problems.
* Chapter 11 — The care and maintenance of your new **COMPUTER**.

Detailed Appendixes provide you with useful charts, comparisons, and special discussions concerning the use and operation of the **COMPUTER**.

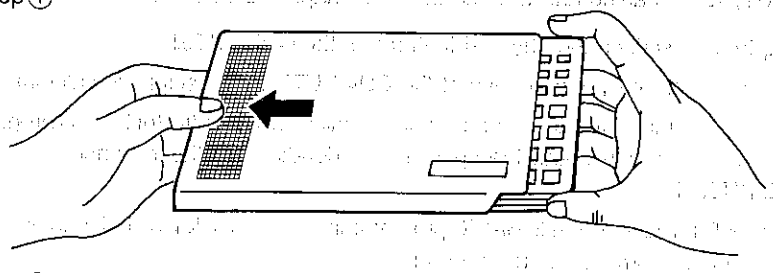
Using the Hard Cover

When the computer is not being used, place the hard (plastic) cover over the operation panel of the computer.

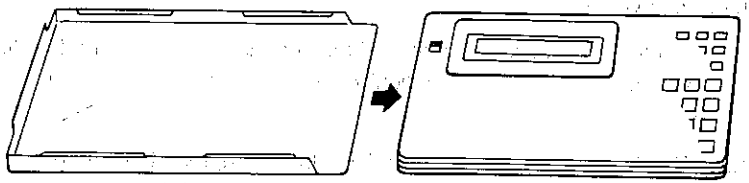
- When the computer is to be used

Remove the hard cover from the computer as shown in figure below.

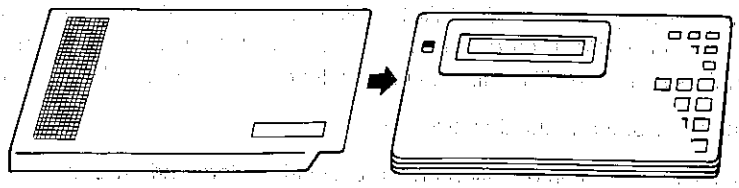
Step ①



Step ②



- When the computer is not used

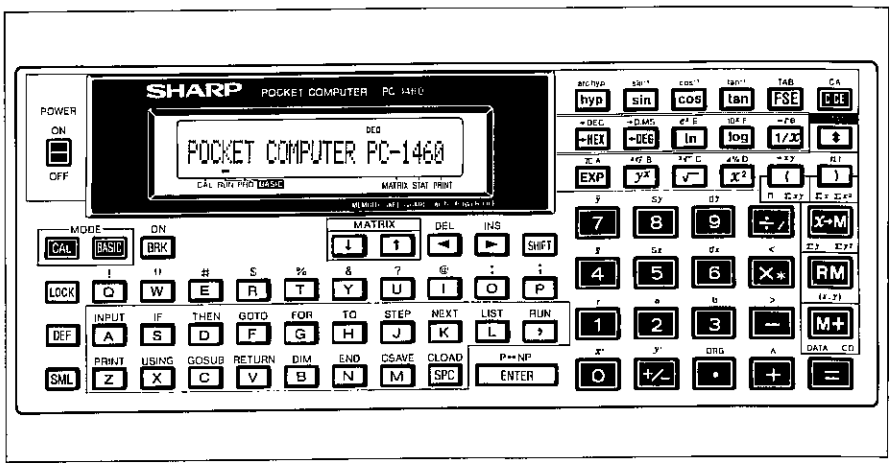


CHAPTER 2
INTRODUCTION TO THE COMPUTER

Description of System

The **SHARP COMPUTER** system consists of:

* 78-character keyboard.
* 24-digit display.
* Powerful BASIC in 72K-byte ROM.
* 8-bit CMOS processor.
* Options: CE-126P Printer/Cassette Interface
  CE-140P Color Dot Printer
  CE-515P Color Plotter/Printer
  1/2-size RAM cards (CE-212M, etc.)
  Regular-size RAM cards (CE-201M, etc.)



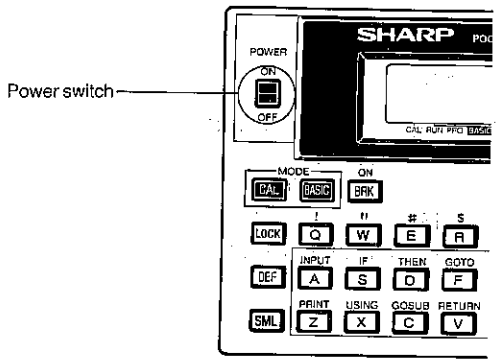
To familiarize you with the placement and functions of parts of the **COMPUTER** keyboard, we will now take up each section of the keyboard. First, just locate the keys and read the description of each. In Chapter 3 we will begin using your new machine.

Key and Switch Operations

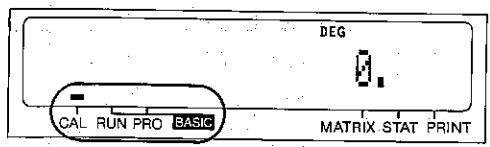
This **COMPUTER** has 78 keys and one slide switch on its panel. Each key function is identified by various letters, numbers, or symbols inscribed on or above the keys.

(1) Power on

To begin with, turn your computer on.
The POWER switch is located at the upper left corner of the computer. Slide the switch to the ON position.



You will see the following initial information in the display:

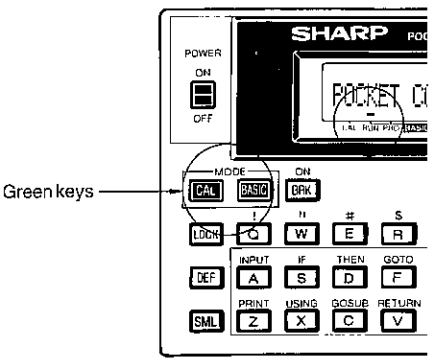


A dash (-) indicator in the lower left area of the display shows the mode in which the computer is now set. When this computer has just been turned on, it functions as a calculator. To show that the computer is in the calculator mode, a dash indicator appears above the CAL (CALculator) label.

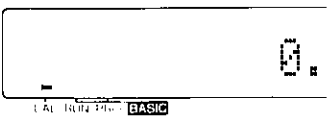
For calculator operations in the CAL mode, refer to CHAPTER 3, USING AS A CALCULATOR.

Modes

The computer can operate basically in three different modes. One mode is the CAL mode, in which you can use your computer just like a calculator.
Another mode is the RUN mode, in which you can execute your program or manual calculation using BASIC commands.
The third mode is the PRO mode, which allows you to store your program into the computer or correct or amend a stored program.
Switching between these modes can be accomplished by the green CAL and BASIC keys. The selected mode is identified with a dash (–) indicator displayed just above the CAL, RUN, or PRO label in the lower left area of the display.



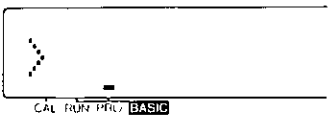
Now switch your computer off, then on again. The CAL mode will be selected.



If you press the BASIC key when in the CAL mode, the RUN mode will be selected.

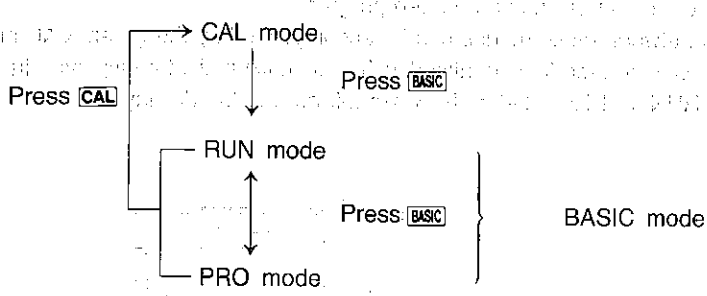


If you press the BASIC key when in the RUN mode, the PRO mode will be selected.



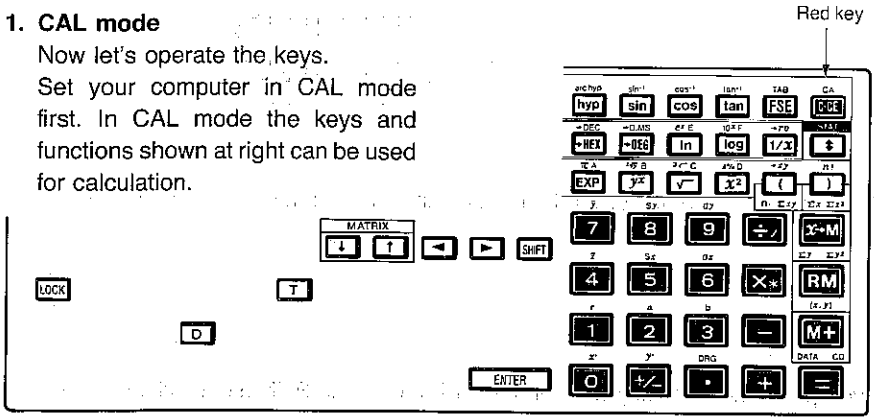
Thus the RUN and PRO modes are selected alternately each time you press the BASIC key.
The computer will return to the CAL mode if you press the CAL key.

Mode switching



1. CAL mode

Now let's operate the keys.
Set your computer in CAL mode first. In CAL mode the keys and functions shown at right can be used for calculation.

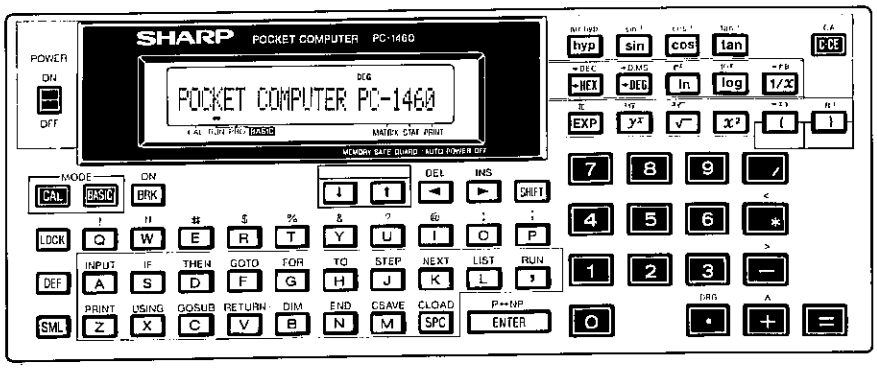


		Display
C·CE (Red key)	→	0.
1 2	→	12.
+	→	12.
3	→	3.
=	→	15.

2. RUN and PRO modes

Change the CAL mode to RUN or PRO by using the [BASIC] key, and press the following keys while watching the display:

In RUN and PRO modes the keys shown below can be used for calculations.



Example:

[PRINT Z] [USING X] [GOSUB C] → ZXC_

[r 1] [a 2] [DRG •] [b 3] → ZXC12.3_

[CA C·CE] → >

[INPUT A] [=] [$\bar{x}$ 4] [$\hat{}$ +] [S_x 5] → A=4 + 5_
↑
Cursor

If you press an alphabet or number key, the item denoted on the key will be entered. When you wish to enter the character or symbol denoted in brown above each key, press the yellow [SHIFT] key before operating the key.

[CA C·CE] [SHIFT] [PRINT Z] → PRINT_

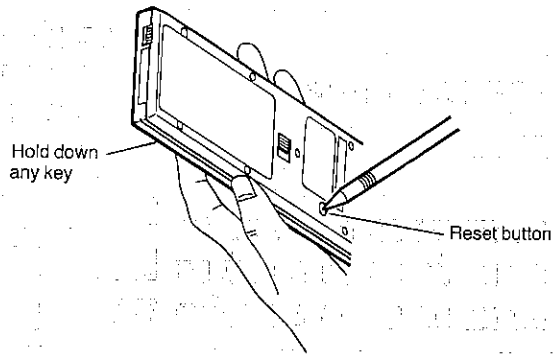
[SHIFT] [" W] [(] [$\sqrt{\ }$] → PRINT " $\sqrt{\ }$ _

The [SHIFT] key is used to enter the characters or symbols inscribed in brown above each key that has two or three functions. If you repeatedly press the [SHIFT] key, the SHIFT symbol at the top of the display will go on and off. The SHIFT symbol indicates that the [SHIFT] key is activated and the characters in brown can be entered.

ALL RESET Button

ALL RESET: Reset button. This button is used to reset the computer when Clear (C-CE) or CA is not sufficient to correct a problem.

To reset the computer, hold down any key on the keyboard and simultaneously press the RESET button on the back. This preserves all programs and variables in memory.



Note: When you press the RESET button, keep pressing it for at least 2 or 3 seconds. If you press the button for a shorter duration, the RESET button may not be activated.
Press the RESET button with a pointed object such as a ballpoint pen. Do not use easily broken points, such as mechanical pencils or the tips of needles, nor points thicker than the hole for the button.

If you get no response from any key even when the above operation is performed, push the RESET button only and the following message will appear on the display.

BUSY
MEMORY ALL CLEAR O.K.?

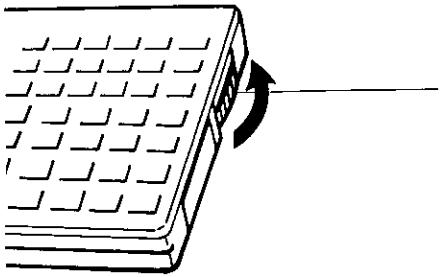
Then press one of the ENTER, Y , and = keys.

Note: If none of the keys is pressed for about 2 minutes after the above display, the COMPUTER is automatically powered off. (See AUTO OFF on page 17.)
This operation will clear all the memory contents (program and data) of the RAM card, so do not press the RESET button without depressing any key unless absolutely necessary.

If the unit still does not operate normally, remove the lithium cells. After waiting about 10 seconds, insert the cells and press the RESET button.

Contrast Control

Your computer has on its right side when viewed from the front, a control for adjusting the contrast of the display. Adjust the display for visibility.



Contrast Control
Turn the control in the arrow direction (counterclockwise) for a higher contrast and in the opposite direction (clockwise) for a lower contrast.

Cell Replacement (Main Power Supply)

The **COMPUTER** normally operates on the two built-in lithium cells.

When replacing the cells, following these cautionary instructions will eliminate many problems:

- Always replace both cells at the same time.
- Do not mix a new cell with a used cell.
- Use only the specified type of lithium cells (CR-2032) (two required).

When to Replace the Cells

If the display is dim and difficult to see when viewed from the front even after the contrast control has been turned counterclockwise as far as it goes, the cell voltage is too low. In this case, replace the cells promptly.

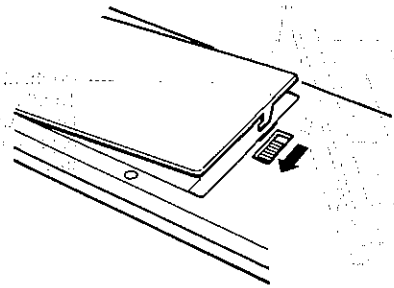
Note: If you are using the optional CE-126P cassette interface and CE-152 cassette tape recorder, save your programs and data in memory onto a cassette tape before replacing the cells.

How to Replace the Cells

(1) Turn off the computer by sliding the power switch to the OFF position.

(2) Remove the back cover from the computer by sliding the lock switch in the arrow direction shown in Figure 1.

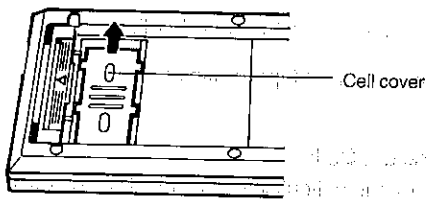
Fig. 1



(3) If there is a RAM card in the card slot, remove it in accordance with the procedure on page 118 or 120.

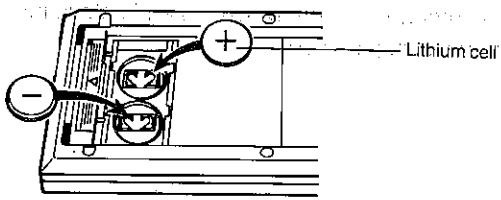
(4) Remove the cell cover by sliding it in the arrow direction shown in the figure below.

Fig. 2



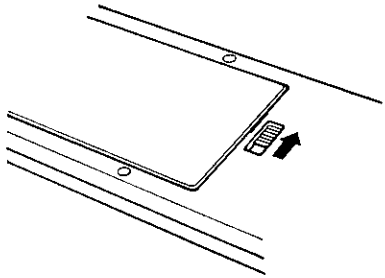
(5) Replace the two cells (Fig. 3).

Fig. 3



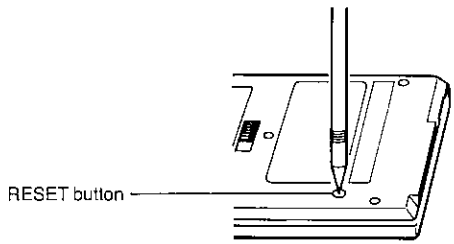
(6) Replace the cell cover by sliding it back in.

(7) Replace the back cover and slide the lock switch to the LOCK position.



Note: Be sure to set the lock switch to the LOCK position, as otherwise the **COMPUTER** will not operate. If the power has been turned on without sliding the lock switch to the LOCK position, slide the lock switch to the LOCK position, turn off the power, and then turn it on again.

(8) Turn on the computer by sliding the power switch to the ON position and press the RESET button to clear the computer.



(9) The display should look like this:



If the display is blank or displays any symbol other than "*", remove the cells and install them again, then check the display.

(10) Turn the power off and insert the RAM card. (See Chapter 8.)

Note: After replacing the cells, be sure to go through above steps (8) and (9) before inserting the RAM card. If the RAM card is installed without turning off the power in step (8), all the RAM card contents will be erased.

RAM CARD Battery Replacement

When to Replace Battery

RAM card contents can be retained for the following periods after inserting a lithium battery (CR-1616) into the RAM card. The lengths of times at a storage temperature of 20°C are specified as follows:

* Approx. 5 years when the card is installed in the computer.
* Approx. 15 months when the card is removed from the computer and kept in storage.

Be sure to install a new battery before these periods expire.

Record the date the battery was replaced on the line provided alongside the battery compartment and be guided accordingly for subsequent battery replacement.

Extreme temperatures, high or low, may decrease battery life, resulting in RAM card content erasure earlier than the above. That the batteries require replacement will be signaled by the following.

(1) Programs in the memory will be inexecutable.
(2) Meaningless data and indications will appear on the display.
(3) Errors will occur frequently and without apparent cause.

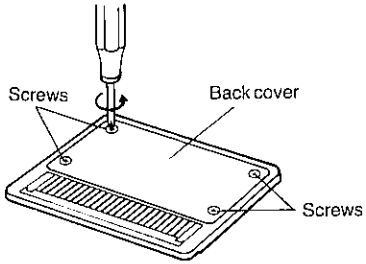
How to Replace Battery

The following is the procedure for replacing the RAM card battery in the optional CE-212M.

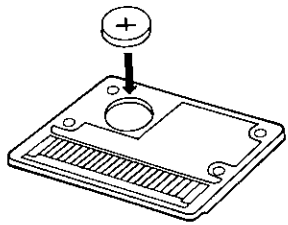
Note: RAM card contents must be recorded onto a tape before replacing the battery, as otherwise, ALL DATA AND PROGRAMS WILL BE LOST WHEN THE BATTERY IS REMOVED FROM THE RAM CARD.

* The battery in any of the optional regular-size RAM cards can be replaced while the card is still in the computer. RAM card contents will be retained by the computer's battery power. (See Chapter 8 for RAM cards.)

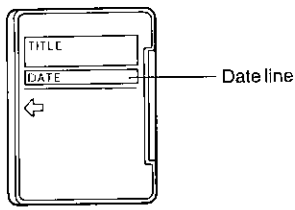
(1) Remove the four screws with a Phillips screwdriver and remove the back cover.



(2) Remove the old battery. Wipe the new battery off with a soft, dry cloth and insert it with its plus side up.



(3) Replace the back cover and tighten the screws.
(4) Record the date of replacement in the DATE line.



* Record the name of the program(s) stored in the RAM card in the TITLE line.
* Reinstall the RAM card into the computer and read in the programs stored on tape, or key in manually.

Note:

- Keeping a dead cell in the computer may result in damage to the computer from solution leakage of the cell. Remove a dead cell promptly.

CAUTION: Keep cell out of reach of children.

CHAPTER 3
USING AS A CALCULATOR

Now that you are familiar with the layout and components of your new **SHARP COMPUTER**, we will begin investigating its exciting capabilities.

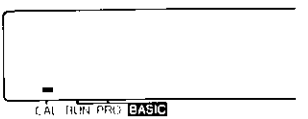
Because the **COMPUTER** allows you the full range of calculating functions, plus the increased power of BASIC programming abilities (useful in more complex calculations), it is commonly referred to as a "smart" calculator. That, of course, makes you a "smart" user!

(Before using the **COMPUTER**, be sure that the two lithium cells supplied as an accessory have been correctly installed.)

Start Up

To turn ON the **COMPUTER**, slide the power switch up.

When you wish to use your **COMPUTER** as a scientific calculator, place the **COMPUTER** in the CAL mode. The CAL mode is selected when the **COMPUTER** is switched on or the [CAL] key is pressed. When the CAL mode has been selected, a dash (-) indicator will appear just above the CAL label in the lower left area of the display



If the dash (-) indicator is above the RUN or PRO label, press the [CAL] key or turn the power off and then on again to select the CAL mode.

Shut Down

To turn off the **COMPUTER**, slide the power switch to the OFF position.

Each time you turn off the machine, the display will be cleared.

Auto OFF

To conserve on battery power, the **COMPUTER** automatically turns off when no keys have been pressed for about 11 minutes. (Note: The **COMPUTER** will not AUTO OFF while you are executing a program.)

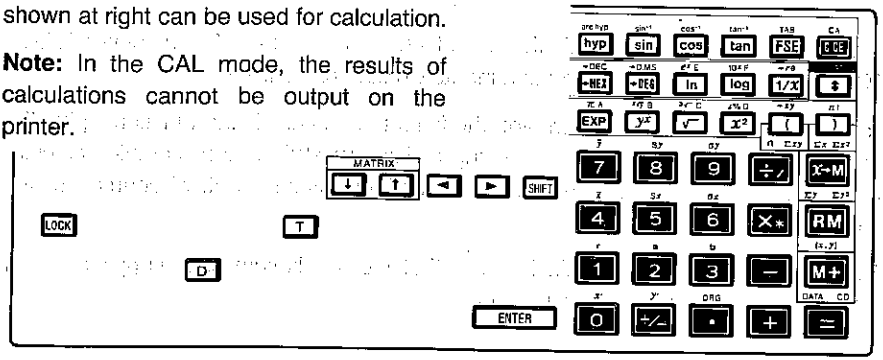
To restart the **COMPUTER** after an AUTO OFF, press the [ON BRK] key located at the right of the green [BASIC] key.

Note that the computer returns to the CAL mode when the ON/BRK key is pressed after the AUTO OFF.

Calculations in the CAL Mode

In the CAL mode, the keys and functions shown at right can be used for calculation.

Note: In the CAL mode, the results of calculations cannot be output on the printer.



Now let us try some simple calculations. Press the following keys while watching the display:

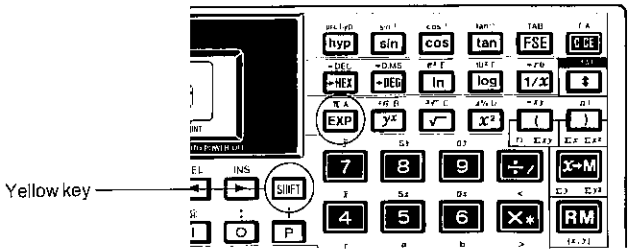
Input	Display
r 1 a 2 b 3	123.
^ +	123.
σx 6 Sx 5 x̄ 4	654.
=	777.
↑ Press the equal key	(123 + 654 = 777)

Did you get the correct answer? If you didn't, turn the computer off, then on again, and try the same calculation.

Now let us call the value of pi (π).

Symbol "π" is inscribed just above the EXP key in brown. The functions identified by brown letters can be used by first pressing the yellow SHIFT key, and then pressing the required function key.

Now press SHIFT EXP .



Input

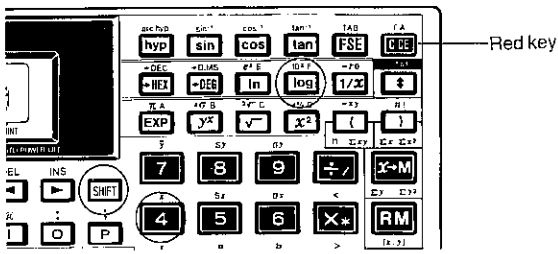
SHIFT π A EXP

Display

3.141592654

($\pi \fallingdotseq 3.141592654$)

What you see in the display is the value of π.
Next, let us compute 10^4. For this calculation, you should use the function 10^x. This function is also identified by a brown letter, so the SHIFT key must be pressed before the function key is pressed:



Input

x̄ 4 SHIFT 10^x F log

Display

10000.

($10^4 = 10000$)

The following outlines the major key functions:

* C·CE (clear) (red key)

If this key is pressed immediately after numeric data is entered or the contents of the memory are recalled, that data will be cleared. In any other case, operation of the C·CE key will clear the operators and/or numeric data that have been entered. The contents of the memory are not cleared with the C·CE key operation.

Input	Display	Input	Display
123 [+] 456	456.	6 [×] 2 [+]	12.
[C·CE]	0.	[C·CE]	0.
789 [=]	912.	6 [÷] 2 [+]	3.
(123 + 789 = 912)		5 [=]	8.

The [C·CE] key may also be used to clear an error.

Input	Display	
5 [÷] 0 [=]	0. E	← Error symbol
[C·CE]	0.	

* [FSE] (display mode switch)

This key is used to switch the display mode for the result of a calculation from the floating point decimal system (normal mode) to the fixed point decimal, scientific notation, or engineering notation system, or vice versa.

Input	Display	
23 [×] 1000 [=]	23000.	(Normal)
[FSE]	FIX 23000.000	(FIX)
[FSE]	SCI 2.300E 04	(SCI)
[FSE]	ENG 23.000E 03	(ENG)

* [TAB] (specifies the number of decimal places)

This key is used to specify the number of decimal places when used in conjunction with a numeral key. Turn off the power switch and then on again. Press [FSE] key and the display will show "0.000" (FIX mode).

	Input	Display
(1) Specifies 2 decimal places.	[SHIFT] [TAB] [FSE] [2]	FIX 0.00 (1)

5 [÷] 8 [=]

FIX	0.63

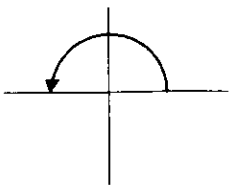
Input	Display
(2) Specifies 5 decimal places. [SHIFT] [TAB FSE] [5]	FIX 0.62500 (2)

* [DRG] (specifies angular unit.)

This key is used to specify the angular units for numeric data used in trigonometric functions, inverse trigonometric functions, or coordinates conversion.

Input	Display	
	DEG	(Degrees)
[SHIFT] [DRG •]	RAD	(Radians)
[SHIFT] [DRG •]	GRAD	(Grads)
[SHIFT] [DRG •]	DEG	(Degrees)

$180^\circ = \pi$ (rad) $= 200^g$



DEG: Degree [°]
RAD: Radian [rad]
GRAD: Grad [g]

* [0] to [9] , [•] , [EXP] and [+/−]

[EXP] : Used to enter a number in exponential form (the display shows "E" following the number entered).

Input	Display
4 [π A] [EXP] 3	4.E Ø3
	(4×10^3)
[=]	4ØØØ.
[+/−]	−4ØØØ.

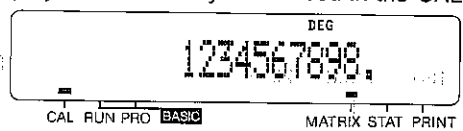
[+/−] : Used to enter a negative number (or to reverse the sign from negative to positive).

Input	Display
1.23 [+/−]	−1.23
[EXP] 5 [+/−]	−1.23E−Ø5
	(-1.23×10^{-5})
[=]	−Ø.ØØØØ123
[+/−]	Ø.ØØØØ123

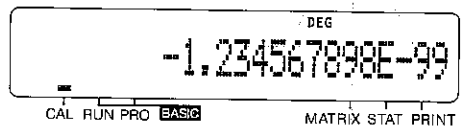
How to Read the Display

This section describes the display formats and symbols used in the CAL mode.

Normal display format



Exponential display format

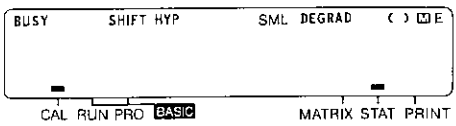


Mantissa (12 digits) Exponent (4 digits)

The computer has a 24-digit display, of which 16 digits are used to display numbers. In the CAL mode, calculation results are normally displayed in the floating decimal point system. If the result is smaller than 0.000000001 or greater than 9999999999 (greater than −0.000000001 or smaller than −9999999999), it is displayed in exponential format. In the exponential format, the mantissa part of a number is displayed to 12 significant digits, while the exponent part is displayed to 4 significant digits (including a decimal point, sign, and symbol).

Display symbols

The following describes the symbols and indicators that appear in the display to show the mode, status, or condition of the computer.



The computer uses the symbols and indicators shown above, whose meanings are the following:

SHIFT: This word comes on when the [SHIFT] or [LOCK] key is activated, indicating that the second function of a key identified by a brown label can be selected. To release the SHIFT mode, press the [SHIFT] or [LOCK] key a second time. To sustain the SHIFT mode, press the [LOCK] key.

HYP: This word comes on when the [hyp] key is pressed, indicating that a hyperbolic function has been selected. If [SHIFT] [hyp] are pressed, a phrase, SHIFT HYP, comes on to indicate that an inverse hyperbolic function has been selected.

SML: This word comes on when the [SML] key is pressed, indicating that the lower case mode for the alphabetic characters is selected.

DEG
RAD
GRAD: These words are selected sequentially each time [SHIFT] [DRG •] keys are operated. Each of these words indicates the angular units for trigonometric functions, inverse trigonometric functions, and coordinates conversion, respectively.

DEG: Degree [°]

RAD: Radian [rad]

GRAD: Grad [g]

(180 deg. = π rad = 200g)

( ): This symbol comes on when parentheses are used in a calculation formula by means of the [(] key.

M: This symbol comes on when a number other than zero is stored in the calculation memory, to indicate that the memory is in use.

E: This symbol comes on if an error has occurred. The error can be cleared by operating the [C·CE] key.

STAT: Pressing [SHIFT] [STAT] keys in the CAL mode causes a dash (-) indicator to appear just above the STAT label in the lower right area of the display. The STAT stands for statistics and indicates that the computer is in the STAT (statistical calculation) mode.

CAL: If a dash (-) indicator appears just above the CAL label in the lower left area of the display, it indicates that the computer is in the CAL (calculation) mode.

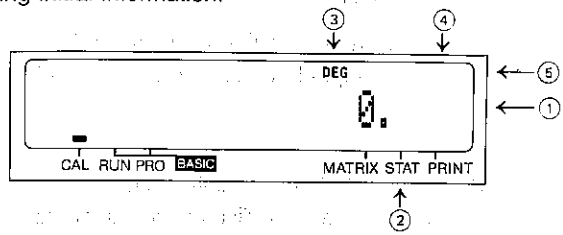
BUSY: This indicator comes on while the computer is performing an arithmetic operation.

MATRIX: Pressing [SHIFT] [↓] or [SHIFT] [↑] in the CAL mode causes a dash indicator (-) to appear above the MATRIX label in the lower right area of the display. The MATRIX indicator indicates that the computer is ready to perform a matrix operation.

To release the MATRIX mode, press either key combination a second time.

Basic Operations

This section describes the basic operations of the computer in the CAL mode. Before starting, turn on the power of your computer. First, press the [CAL] key to place the computer in the CAL mode. Then press [C·CE] [C·CE] , and make sure that the display shows the following initial information.



If not, read the following description and take the necessary action:

① More than one zero is displayed (e.g., 0.00):
The number of fractional digits is being specified. Clear the TAB setting by turning off the power switch then on again. The COMPUTER is now in the normal display mode.

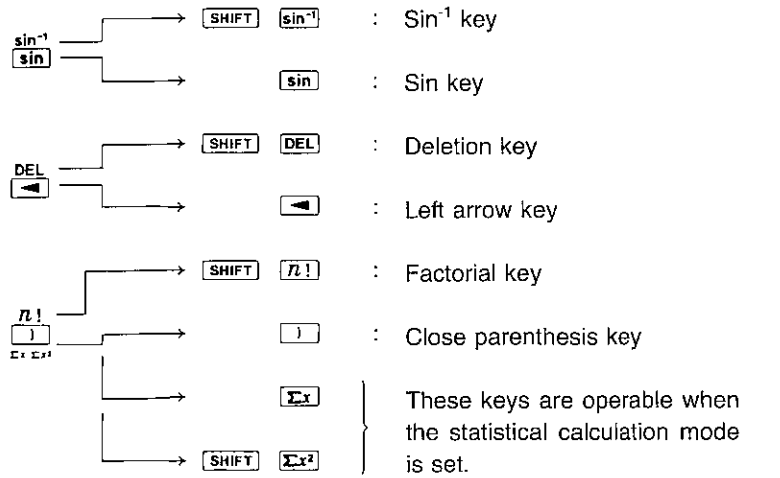
② A dash (-) indicator is displayed at the STAT or MATRIX label:
The computer is in the statistical calculation mode. Press [SHIFT] [STAT] to release the STAT mode. Press [SHIFT] [↓] or [SHIFT] [↑] to release the MATRIX mode.

③ RAD or GRAD is displayed instead of DEG:
The RAD, GRAD, and DEG indicate angular units for display data. Any of these symbols may be displayed unless a trigonometric function, inverse trigonometric function, or coordinate conversion is to be executed. Each of these symbols can be sequentially selected by operating [SHIFT] [DRG] .

④ Symbol [M] is displayed:
Numeric data is already in the memory. This symbol can be cleared by [C·CE] [x→M] .

⑤ All symbols displayed in the upper area of the display can be cleared with the [C·CE] key, with the exception of those described in the above items ③ and ④.

In this manual, the key functions are shown as follows:



1. Addition, Subtraction

Key in the following: 12 [+] 45.6 [−] 32.1 [+] 789 [−] 741 [+] 213 [=]

Answer: 286.5

2. Multiplication, Division

a. Key in the following: 841 [×] 586 [÷] 12 [=]

Answer: 41068.83333

b. Key in the following: 427 [+] 54 [×] 32 [÷] 7 [−] 39 [×] 2 [=]

Answer: 595.8571429

Note that multiplication and division have priority over addition and subtraction. In other words, multiplication and division will occur before addition and subtraction.

Constant Multiplication: The first number entered is a constant.

Key in: 3 [×] 5 [=] Answer: 15

Key in: 10 [=] Answer: 30

Constant Division: The number entered after the division sign is a constant.

Key in: 15 [÷] 3 [=] Answer: 5

Key in: 30 [=] Answer: 10

Note: The machine places some calculations in pending status depending on their priority levels. Accordingly, in successive calculations the operator and numerical value of the calculation last performed in the computer are handled as a calculating instruction and a constant for the next calculation, respectively.

$a + b \times c =$	$+bc$	(Constant addition)
$a \times b \div c =$	$\div c$	(Constant division)
$a \div b \times c =$	$\frac{a}{b} \times$	(Constant multiplication)
$a \times b - c =$	$- c$	(Constant subtraction)

3. Memory Calculations

The independently accessible memory can be accessed by using the three keys: [x→M], [RM], [M+]. Before starting a calculation, clear the memory by pressing [C·CE] and [x→M].

Key in: 12 [+] 5 [M+] Answer: 17

To subtract, key in: 2 [+] 5 [=] [+/−] [M+]

Answer to this equation: −7

Key in [RM] to recall memory: 10 is displayed.

Key in: 12 [×] 2 [=] [x→M]

Answer: 24 (Also takes place of 10 in memory)

Key in: 8 [÷] 2 [M+]

Answer: 4 [RM] : 28

Note: Memory calculations are impossible in the STAT (Statistical calculation) mode.

When subtracting a number from the memory, press the [+/−] and [M+] keys.

Scientific Calculations in the CAL mode

To perform trigonometric or inverse trigonometric functions, and coordinates conversion, designate the angular unit for the calculation. The angular unit "DEG, RAD, or GRAD" is designated by the [SHIFT] and [DRG] keys.

1. Trigonometric functions

Set the angular unit to "DEG".
Calculate: Sin 30° + Cos 40° =
Key in the following: 30 [sin] + 40 [cos] [=]
Answer: 1.266044443
Calculate: Cos 0.25π
Set the angular unit to "RAD".
Key in: .25 [×] [SHIFT] [π] [=] [cos] (Remember to use the [SHIFT] key.)
Answer: 0.707106781

2. Inverse Trigonometric Functions:

Calculate: $\sin^{-1}$ 0.5
Set the angular unit to "DEG".
Key in: .5 [SHIFT] [sin⁻¹] Answer: 30
Calculate: $\cos^{-1}$ −1
Set the angular unit to "RAD".
Key in: 1 [+/−] [SHIFT] [cos⁻¹] To enter a negative number, press the [+/−] key after a number.
Answer: 3.141592654 (Value of π)

The calculation results of the respective inverse trigonometric functions will be displayed within the following limits.

$\theta=\sin^{-1}\chi$, $\theta=\tan^{-1}\chi$	$\theta=\cos^{-1}\chi$
DEG: $-90\leqq\theta\leqq90$ [°]	DEG: $0\leqq\theta\leqq180$ [°]
RAD: $-\pi/2\leqq\theta\leqq\pi/2$ [rad]	RAD: $0\leqq\theta\leqq\pi$ [rad]
GRAD: $-100\leqq\theta\leqq100$ [g]	GRAD: $0\leqq\theta\leqq200$ [g]

3. Hyperbolic and Inverse Hyperbolic Functions

Calculate: Sinh 4
Key in: 4 [hyp] [sin] Answer: 27.2899172
Calculate: Sinh^{-1} 9
Key in: 9 [SHIFT] [arc hyp] [sin] Answer: 2.893443986

4. Power Functions

Calculate: 20^2
Key in: 20 [x^2] Answer: 400

Calculate: 3^3 and 3^4
Key in: 3 [y^x] 3 [=] Answer: 27
Key in: 3 [y^x] 4 [=] Answer: 81

5. Roots

Calculate: $\sqrt{25}$
Key in: 25 [$\sqrt{\ }$] Answer: 5
Calculate: Cubic root of 27
Key in: 27 [SHIFT] [$\sqrt[3]{\ }$] Answer: 3
Calculate: Fourth root of 81
Key in: 81 [SHIFT] [$\sqrt[x]{y}$] 4 [=] Answer: 3

6. Logarithmic Functions

Calculate: ln 21, log 173
Natural Logarithms:
Key in: 21 [ln] Answer: 3.044522438
Common Logarithms:
Key in: 173 [log] Answer: 2.238046103

7. Exponential Functions

Calculate: $e^{3.0445}$
Key in: 3.0445 [SHIFT] [e^x]
Answer: 20.99952881 (21 as in item "6" above)
Calculate: $10^{2.238}$
Key in: 2.238 [SHIFT] [10^x]
Answer: 172.9816359 (173 as in item "6" above)

8. Reciprocals

Calculate: 1/6 + 1/7
Key in: 6 [$1/x$] [+] 7 [$1/x$] [=] Answer: 0.309523809

9. Factorial

Calculate: 69!
Key in: 69 [SHIFT] [$n!$]
Answer: 1.711224524E 98 ($=1.711224524\times10^{98}$)

Note that the section on Errors deals with the calculation limits of the computer.

Calculate: $_8P_3 = \dfrac{8!}{(8-3)!} =$
Key in: 8 [SHIFT] [$n!$] [÷] [(] 8 [−] 3 [)] [SHIFT] [$n!$] [=]
Answer: 336

10. Percent calculations

Calculate: 45% of 2,780 ($2{,}780\times\frac{45}{100}$)
Key in: 2780 [×] 45 [SHIFT] [Δ%] Answer: 1251

Calculate: $\frac{547 - 473}{473} \times 100$

Key in: 547 [−] 473 [SHIFT] [Δ%]

Answer: 15.6448203

11. Angle/Time conversions

To convert an angle given in the sexagesimal system (degrees/minutes/seconds) to its decimal equivalent, a value in degrees must be entered as an integer and values in minutes and seconds as decimal fractions, respectively.

Convert 12°47′ 52″ to its decimal equivalent.

Key in: 12.4752 [→DEG]

Answer: 12.79777778

When converting an angle in decimal degrees to its sexagesimal equivalent (degrees/minutes/seconds), the answer is broken down: integer part = degrees; 1st and 2nd decimal digits = minutes; 3rd and 4th digits = seconds; and the 5th digit and up = fractional seconds.

Convert 24.7256 to its sexagesimal equivalent (degrees/minutes/seconds)

Key in: 24.7256 [SHIFT] [→D.MS]

Answer: 24.433216 or 24°43′32″

A racehorse has the track times of 2 minutes 25 seconds, 2 minutes 38 seconds, and 2 minutes 22 seconds. What is the average running time of the horse?

Key in: .0225 [→DEG] [+] .0238 [→DEG] [+] .0222 [→DEG] [=]

Answer 1: 0.123611111

Key in: [÷] 3 [=]

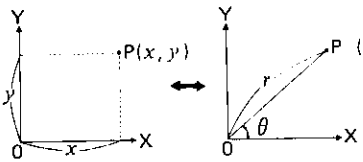
Answer 2: 0.041203703

Key in: [SHIFT] [→D.MS]

Answer 3: 0.022833333 or the average time is 2 minutes 28 seconds

12. Coordinates Conversion

Converting rectangular coordinates to polar (x, y → r, θ)



$r = \sqrt{x^2 + y^2}$

$\theta = \tan^{-1} \frac{y}{x}$

DEG: $0 \leqq |\theta| \leqq 180$

RAD: $0 \leqq |\theta| \leqq \pi$

GRAD: $0 \leqq |\theta| \leqq 200$

Solve for χ = 6 and y = 4

Angular unit: DEG

Key in: 6 [↕] 4 [SHIFT] [→r∠θ] Answer: 7.211102551 (r)

Key in: [↕] Answer: 33.69006753 (θ)

Calculate the magnitude and direction (phase) in vector i = 12 + j9

Key in: 12 [↕] 9 [SHIFT] [→r∠θ] Answer: 15 (r)

Key in: [↕] Answer: 36.86989765 (θ)

Converting polar coordinates to rectangular (r, θ → x, y)

Solve for P (14, π/3), r = 14, θ = π/3)

Angular unit: RAD

Key in: [SHIFT] [π] [÷] 3 [=] [↕] 14 [↕] [SHIFT] [•xy]

Answer: 7.000000002 (x)

Key in: [↕]

Answer: 12.212435565 (y)

In the above example, θ = π/3 is input first and is replaced with r = 14 by pushing the [↕] key after r is input.

Use of Parenthesis

The parentheses keys are needed to cluster together a series of operations when it is necessary to override the priority system of algebra. When parentheses are in use on the **COMPUTER**, the symbol "( )" will appear in the display. Calculations in parentheses have priority over other calculations. Parentheses in the CAL mode can be used up to 15 times in a single level. A calculation within the innermost set of parentheses will be performed first.

Calculate: 12 + 42 ÷ (8 − 6)

Key in: 12 [+] 42 [÷] [(] 8 [−] 6 [)] [=]

Answer: 33

Calculate: 126 ÷ [(3 + 4)× (3 − 1)]

Key in: 126 [÷] [(] [(] 3 [+] 4 [)] [×] [(] 3 [−] 1 [)] [)] [=]

Answer: 9

Note: The [)] keys located just before the [=] or [M+] key can be omitted.

Decimal Places

The [SHIFT] and [TAB] keys are used to specify the number of decimal places in the calculation result. The number of decimal places after the decimal point is specified by the numeral key ([Ø] ~ [9]) pressed after the [SHIFT] and [TAB] keys. In this case, the display mode must be FIX (fixed decimal point), SCI (scientific notation), or ENG (engineering notation).

[SHIFT] [TAB] [Ø] → Designates 0 decimal place.
(The 1st decimal place is rounded.)

[SHIFT] [TAB] [1] → Designates 1 decimal place.
(The 2nd decimal place is rounded.)

[SHIFT] [TAB] [9] → Designates 9 decimal places.
(The 10th decimal place is rounded.)

To clear the TAB setting (designation of the decimal places), turn off the power switch and then on again. The display is now in the normal display mode.

Example:

[SHIFT] [TAB] [9] → 0.055555556 (FIX mode)
[.] [5] [÷] [9] [=] (The 10th decimal place is rounded.)

[FSE] → 5.555555556E-02 (SCI mode)
(The 10th decimal place of the mantissa part is rounded.)

[SHIFT] [TAB] [3] → 5.556E-02 (SCI mode)
(The 4th decimal place of the mantissa part is rounded.)

[FSE] → 55.556E-03 (ENG mode)

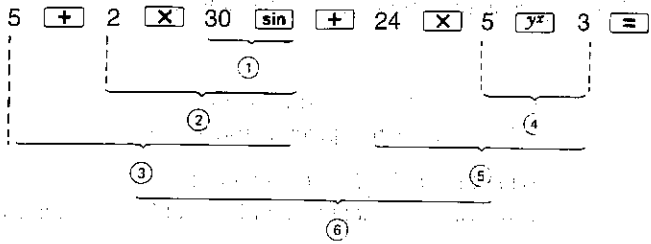
[FSE] → 0.055555555
This is determined by the computer in the form of $5.55555555555 \times 10^{-2}$. Rounding the 11th digit of the mantissa results in $5.555555556 \times 10^{-2}$. When changed to the floating decimal point display, the rounded part may not be displayed as in this example.

Priority Levels in CAL Mode

The machine is provided with a function that judges the priority levels of individual calculations, which permits keys to be operated according to a given mathematical formula. The following shows the priority levels of individual calculations.

Level	Operations
(1)	Functions, such as sin, x^2
(2)	y^x, $\sqrt[x]{y}$
(3)	x, ÷ (Calculations which are given the same priority level are executed in their sequence of input.)
(4)	+, −
(5)	=, M+, Δ%

Ex. Key operation and sequence of calculation in $5 + 2 \times \sin 30 + 24 \times 5^3 =$



The numbers ① ~ ⑥ indicate the sequence in which the calculations are carried out.

When calculations are executed from the higher priority one in sequence, a lower priority one must be set aside. The machine is provided with a memory area for up to eight levels of pending operations.
As the memory area can also be used in a calculation including parentheses, calculations can be performed according to a given mathematical formula unless the levels of parentheses and/or pending operations exceed 8 in total.

- Single-variable functions are calculated immediately after key operation without being retained. (x^2, $1/x$, n!, →DEG, →DMS, etc.)

Calculation without using parentheses

Ex. 1 + 2 =
① — Pending of 1 level

1 + 2 × 3 =
① ② — Pending of 2 levels

1 + 2 × 3 y^x 4 =
① ② ③ — Pending of 3 levels

1 + 2 × 3 y^x 4 ÷ 5
① ③ ② — With the y^x pressed, 3 calculations remain pending. Pressing the ÷ key executes the calculations of "y^x" highest in priority level and "x" identical in priority level. After the ÷ key is pressed, the other 2 calculations will remain pending.

Calculation using parentheses

Ex. i) 1 [+] 2 [×] 3 [y^x] [(]

(1) (2) (3)

4 [÷] 5

(4)

4 numerals and calculation instructions are left pending.

ii) 1 [+] 2 [×] [(] 3 [−]

① ② ③

4 [÷] 5 [)]

④

Pressing the [)] key executes the calculation of 3 − 4 ÷ 5 in the parentheses, leaving 2 calculations pending.

- Parentheses can be used unless pending calculations exceed 8. However, parentheses can be continuously used up to 15 times.

Ex. a×(((b − c×(((d + e)×f) ÷ g .....

Parentheses, if continued, can be used up to 15.

Conversion between Decimal and Hex Numbers, and Hex Calculations

([→HEX] , [→DEC])

[→HEX] : Allows you to convert a decimal number into its hexadecimal equivalent and, at the same time, places the computer in the HEX mode. (The display shows the symbol "HEX".)

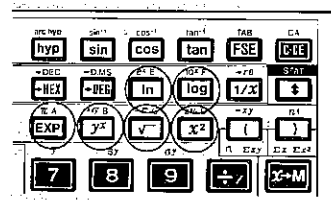
[SHIFT] [→DEC] : Allows you to convert a hexadecimal number into its decimal equivalent and, at the same time, releases the computer from the HEX mode. (Symbol "HEX" disappears from the display.)

Hexadecimal notation is one of the notation systems broadly used in the computer field. The radix for hex notation is 16 and hex numbers consist of numerals 0 through 9 and uppercase letters A through F used in place of 10 through 15 of decimal notation.

(Hexadecimal)		(Decimal)
A	———	10
⟨		⟨
F	———	15

Hex numbers A through F can be entered by first placing your computer in the Hex mode (with [•HEX] key), then pressing the respective keys shown in figure.

The symbol HEX indicates that the numeric data shown in the display is a hex number, and that you can perform any basic arithmetic operations on hex numbers.



To clear the Hex mode, operate [SHIFT] [•DEC] . You cannot clear it with the [C·CE] key.

1. Decimal to hex conversion

Example: Convert decimal number 30 into its hexadecimal equivalent:

Key in: 30 [•HEX] Answer: 1E. HEX

To perform a new conversion, temporarily clear the HEX mode with [SHIFT] [•DEC] .

Example: Convert decimal number −2, into its hexadecimal equivalent. Temporarily clear the HEX mode with [C·CE] [SHIFT] [•DEC] .

Key in: [2] [+/−] [•HEX]

Answer: FFFFFFFFE. HEX

- If you attempt decimal-to-hex conversion on a negative decimal number, the computer internally performs "two's complement" calculation and shows the result in 16's complement.
- The [+/−] key may be used to reverse the positive or negative sign of the numeric data now in the display. If the sign of a positive hex number is reversed, the complement of the positive number will be obtained in the display.

Example: Convert decimal number 123.4 into its hexadecimal equivalent.

Key in: [SHIFT] [•DEC] 123.4 [•HEX]

Answer: 7B. HEX

- If a decimal number having a fractional part is converted into a hex number, the fractional part of the decimal number is truncated and only its integer part is converted into a hex number.

2. Hex to decimal conversion

Example: Convert hex number 2BC into its decimal equivalent
Key in: [C·CE] [→HEX] 2 B C [SHIFT] [→DEC]

Answer: 700.

Example: Convert hex number FFFFFFFF12 into its decimal equivalent:
Key in: [C·CE] [→HEX] FFFFFFFF 12 [SHIFT] [→DEC]

Answer: FFFFFFFF12. HEX

-238.

- If any of hex numbers FFFFFFFFFF to FDABF41C01 is converted into its decimal equivalent, the corresponding decimal number will become negative.

3. Hexadecimal calculations

Hexadecimal calculations can be done after your computer is placed in the Hex mode. Press [C·CE] [→HEX] and the symbol HEX will be displayed.

Example: A4 + BA =
Key in: A4 [+] B A [=]

Answer: 15 E. HEX

(350 in decimal)

Example: 8×3 =
Key in: 8 [×] 3 [=]

Answer: 18. HEX

(24 in decimal)

Example: (12 + D)×B =
Key in: [C·CE] [(] 12 [+] D [)] [×] B [=]

Answer: 155. HEX

(341 in decimal)

Example: 43A − 3CB =
+)A38 − 2FB =
Total

Key in: [C·CE] [Σ→M]
4 3 A [−] 3 C B [M+]

Answer: 6F. HEX

A 3 8 [−] 2 F B [M+]

Answer: 73D. HEX

[RM]

Answer: 7AC. HEX

For hex calculations, you should note the following points:

- In hex calculations, the computer ignores all fractional parts. This means that the decimal point key, [•] , is meaningless even if pressed for a hex calculation.
- If an intermediate result in successive hex calculations includes a fractional part, an error will result.

  Example: B [÷] 3 [×] ... Error (Symbol "E" is displayed.)

  If a fractional part is in the result of the final calculation, it will be truncated and only the integer part of the result will be displayed.

  Example: B [÷] 3 [=] ... 3. HEX

- In the Hex mode, the [+/−] key may be used to obtain a complement for the hex number now shown in the display.

  Example: A B [+/−] → FFFFFFFF55. HEX

  [+/−] → AB. HEX

- In the Hex mode, the function keys on the computer are not usable.
- When the computer is in the STAT or MATRIX mode (a dash (-) indicator is shown at the STAT or MATRIX label), neither conversion between decimal and hex numbers nor a hex calculation is executable.

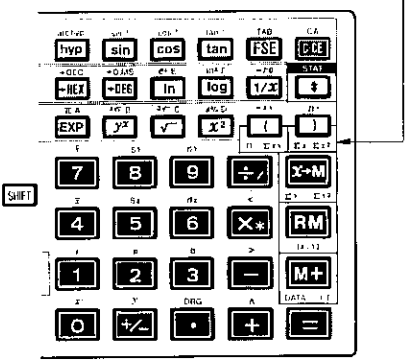
Statistical Calculations

To perform statistical calculation, press the [SHIFT] and [STAT] keys (under the red [C·CE] key) in the CAL mode, a dash (-) indicator will appear just above the "STAT" label in the lower right area of the display. The "STAT" stands for STATistics, and indicates that the computer is in the statistical calculation mode.
When the computer is in the RUN or PRO mode, press the [CAL] and then [SHIFT] [STAT] to perform a statistical calculation.



Display a dash indicator in this position by pressing the [SHIFT] and [STAT] keys.

Keys that are used mainly in the statistical calculation mode.



When a statistical calculation is performed, the following statistics are automatically stored in the memory area for fixed variables used in the BASIC mode. And these statistics can be used in the BASIC mode, because these statistics are retained even when the statistical calculation mode is reset. These statistics are cleared when the statistical calculation mode is reset and then set again for another statistical calculation.

Memory	Z	Y	X	W	V	U
Statistic	n	Σx	Σx^2	Σxy	Σy	Σy^2

To clear previous statistical inputs and calculations, reset the statistical calculation mode once and set this mode again. Otherwise, when a new statistical calculation is performed, incorrect answers will be obtained.

When the statistical calculation mode is set, the following cannot be performed:

* Memory calculation
* Calculation with parentheses
* Coordinates conversion
* Conversion between hexadecimal and decimal numbers
* Hexadecimal calculation

1. Single-variable Statistical Calculation

The following statistics are obtainable in a single-variable statistic calculation:

(1) n: Number of samples

(2) Σx: Sum total of samples

(3) Σx^2: Sum of squares of samples

(4) $\bar{x}$: Mean value of samples $\bar{x} = \dfrac{\Sigma x}{n}$

(5) Sx: Standard deviation with population parameter taken to be "n−1".

$$Sx = \sqrt{\frac{\Sigma x^2 - n\bar{x}^2}{n-1}}$$

(Used to estimate the standard deviation of a population from the sample data extracted from that population.)

(6) σx: Standard deviation with population parameter taken to be "n".

$$\sigma x = \sqrt{\frac{\Sigma x^2 - n\bar{x}^2}{n}}$$

(Used when all the populations are taken as sample data or when finding the standard deviation of a population with samples taken as that population.)

Data for single-variable statistic calculations are input by the following key operations:

(1) Data [DATA] (used to enter data one by one)

(2) Data [×] Frequency [DATA] (used to enter two or more of the same data)

Example:

Calculate standard deviation, mean, and variance $(Sx)^2$ from the following data: Set the computer in the statistical calculation mode.

Value	35	45	55	65
Frequency	1	1	5	2

As each sample is entered, the number of data of that sample will appear at the right of the display.

	Key in:	Display
	[SHIFT] [STAT]	0.
35	[DATA]	1.
45	[DATA]	2.
55×5	[DATA]	7.
65×2	[DATA]	9.

Notes: 1. After all the data have been entered, statistics such as mean value, standard deviation, etc., may be obtained in any desired order.

2. After a mean value, standard deviation, or any other statistic has been obtained as an intermediate result, more data can be entered and statistical calculations can be performed continuously on additional data entry.

	Key in:	Display:
Mean:	[SHIFT] [$\bar{x}$]	53.88888889
Standard Deviation:	[SHIFT] [S_x]	9.279607271
Variance:	[x^2]	86.11111111

Correct Data (CD): The last data entry in the above example is an error and must be changed to 60×2.

Key in:	Display:
65 [×] 2 [SHIFT] [CD]	7.
60 [×] 2 [DATA]	9.

2. Two-variable Statistics and Linear Regression

In addition to the statistics for both variables x and y which are the same as those of x in single-variable statistics, the sum of the products of samples Σxy is obtained in two-variable statistics. Two-variable statistics make possible the development of a relationship (correlation) between two sets of data. Each pair of data has x and y values. From these sets of data a line of regression can be established. The relationship of the two sets of data by use of the straight line method is called Linear Regression. In Linear Regression there are three important values, r, a, and b.

The equation of the straight line is $y = a + bx$, where a is the point at which the line crosses the Y-axis and b is the slope of the line.

The correlation coefficient r shows the relationship between two sets of data. A perfect correlation between two values is an r equal to 1 (−1 is a perfect negative correlation); in other words, by knowing the value of one variable you can predict with 100% accuracy the value of the other variable. The further the value of r is from 1, the less reliable will your predictions be. The following table can be used as a set of definitions of the values of the correlation coefficient:

	Value of r	Call it
Positive Correlation	+0.80 to +1.00	Extra High
	+0.60 to +0.80	High
	+0.40 to +0.60	Moderate
	+0.20 to +0.40	Low
	−0.20 to +0.20	Nil
Negative Correlation	−0.20 to −0.40	Low
	−0.40 to −0.60	Moderate
	−0.60 to −0.80	High
	−0.80 to −1.00	Extra High

r: Correlation coefficient

$$r = \frac{S_{xy}}{\sqrt{S_{xx} \cdot S_{yy}}}$$

a: $a = \bar{y} - b\bar{x}$

b: $b = \dfrac{S_{xy}}{S_{xx}}$

Coefficient of linear regression equation $y = a + bx$

$$S_{xx} = \Sigma x^2 - \frac{(\Sigma x)^2}{n}$$

$$S_{yy} = \Sigma y^2 - \frac{(\Sigma y)^2}{n}$$

$$S_{xy} = \Sigma xy - \frac{\Sigma x \cdot \Sigma y}{n}$$

Example 1: If we know a student's mark in mathematics, can we predict the mark in English?

The exam marks for five students chosen at random are given in the following table:

Student No. n	Mark in Math. x	Mark in English y
1	82	79
2	53	50
3	61	87
4	74	96
5	51	73
6	51	73

Key in:	Display
82 (x,y) 79 DATA	1.
53 (x,y) 50 DATA	2.
61 (x,y) 87 DATA	3.
74 (x,y) 96 DATA	4.
51 (x,y) 73 × 2 DATA	6. (Note: To input multiple identical samples, proceed as indicated.)

SHIFT r	0.571587901
SHIFT a	34.26190476 (y-axis)
SHIFT b	0.678571428 (slope)

The value of 0.571587901 for r indicates that the correlation is moderate. The equation for the straight line for this data is $y = 34.26 + 0.68x$ when truncated to second decimal places.

Key in:	Display
90 SHIFT y'	95.33333333

If we had a student whose mark in mathematics was 90, the student would have a mark of 95 in English based on this analysis.

Example 2: Is weight a good predictor of longevity among men 65 years of age? In 1950, 10 men, each six feet tall, were selected for an experiment to determine if their weight effected their life span.

Sample	1	2	3	4	5	6	7	8	9	10
Age at death	72	67	69	85	91	68	77	74	70	82
Weight at age 65	185	226	200	169	170	195	175	174	198	172

Key in:	Display
SHIFT STAT	0.
72 (x,y) 185 DATA	1.
67 (x,y) 226 DATA	2.
(Continue to place in all data)	⋮
SHIFT r	−0.792926167

The value for r indicates a relatively high negative correlation. A higher weight means a shorter life span. To graph the regression line, coefficients a and b are used.

SHIFT a	321.9292125 (y-axis)
SHIFT b	−1.795088908 (slope)

Predict the age of death of a 6-foot man weighing 190 pounds in 1950.

190 SHIFT x' 73.4945283 years

To reach age 90, what should a man's weight be in 1960?

90 SHIFT y' 160.3712108 pounds

To reach age 150, what should a man's weight be? Obviously, the answer will make no sense, indicating the danger of carrying a straight-line extrapolation too far.

CAUTION

The following statistical data obtained in the CAL mode can be used in the BASIC mode.

Memory	Z	Y	X	W	V	U
Statistic	n	Σx	Σx^2	Σxy	Σy	Σy^2

When performing calculations using this statistical data, use the RUN mode.

For example, to determine the sum of squares (S^2) of four pieces of data, 205, 221, 226, and 220, operate your computer as follows:

$$S^2 = \Sigma (x - \bar{x})^2$$
$$= \Sigma x^2 - n\bar{x}^2$$
$$= \Sigma x^2 - \frac{1}{n}(\Sigma x)^2$$

- Enter the data in the CAL mode.

[CAL] [SHIFT] [STAT]

```
0.
```

205 [DATA] 221 [DATA]

226 [DATA] 220 [DATA]

```
4.
```

- Change the CAL mode to RUN and calculate S^2.

[BASIC]

```
>
```

[X] [−] [Y] [X*] [Y] [÷/] [Z]

```
X-Y*Y/Z_
```

[ENTER]

```
246.
```

Calculation Range

Four arithmetic calculations:

1st operand, 2nd operand, and
calculation result: $\pm 1 \times 10^{-99} \sim \pm 9.999999999 \times 10^{99}$ and 0

Scientific functions:

Functions	Dynamic range	Note
$\sin x$ $\cos x$ $\tan x$	DEG: $\lvert x \rvert < 1 \times 10^{10}$ RAD: $\lvert x \rvert < \frac{\pi}{180} \times 10^{10}$ GRAD: $\lvert x \rvert < \frac{10}{9} \times 10^{10}$ In $\tan x$, however, the following cases are excluded. DEG: $\lvert x \rvert = 90 (2n - 1)$ RAD: $\lvert x \rvert = \frac{\pi}{2} (2n - 1)$ n = integer GRAD: $\lvert x \rvert = 100 (2n - 1)$	
$\sin^{-1} x$ $\cos^{-1} x$	$-1 \leqq x \leqq 1$	
$\tan^{-1} x$	$\lvert x \rvert < 1 \times 10^{100}$	
$\ln x$ $\log x$	$1 \times 10^{-99} \leqq x < 1 \times 10^{100}$	($\ln x = \log_e x$)
e^x	$-1 \times 10^{100} < x \leqq 230.2585092$	($e \fallingdotseq 2.718281828$)
10^x	$-1 \times 10^{100} < x < 100$	
y^x (^)	• $y > 0$: $-1 \times 10^{100} < x \log y < 100$ • $y = 0$: $x > 0$ • $y < 0$: x: integer or $\frac{1}{x}$: odd number $-1 \times 10^{100} < x \log \lvert y \rvert < 100$	$y^x = 10^{x \cdot \log y}$
$\sqrt[x]{y}$	• $y > 0$: $-1 \times 10^{100} < \frac{1}{x} \log y < 100$, $x \neq 0$ • $y = 0$: $x > 0$ • $y < 0$: x or $\frac{1}{x}$: integer ($x \neq 0$) $-1 \times 10^{100} < \frac{1}{x} \log \lvert y \rvert < 100$	$\sqrt[x]{y} = 10^{\frac{1}{x} \cdot \log y}$
$\sqrt[3]{x}$	$\lvert x \rvert < 1 \times 10^{100}$	
$\sinh x$ $\cosh x$ $\tanh x$	$-227.9559242 \leqq x \leqq 230.2585092$	
$\sinh^{-1} x$	$\lvert x \rvert < 1 \times 10^{50}$	
$\cosh^{-1} x$	$1 \leqq x < 1 \times 10^{50}$	
$\tanh^{-1} x$	$\lvert x \rvert < 1$	

Functions		Dynamic range	Note
$\sqrt{x}$		$0 \leqq x < 1 \times 10^{100}$	
x^2		$\lvert x \rvert < 1 \times 10^{50}$	
$\frac{1}{x}$		$\lvert x \rvert < 1 \times 10^{100}$ $x \neq 0$	
n!		$0 \leqq n \leqq 69$ (n: Integer)	
D.MS→DEG		$\lvert x \rvert < 1 \times 10^{100}$	
DEG → D.MS		$\lvert x \rvert < 1 \times 10^{100}$	
HEX → DEC		$0 \leqq x \leqq$ 2540BE3FF FDABF41C01 $\leqq x \leqq$ FFFFFFFFFF	x is an integer in HEX mode
DEC → HEX		$\lvert x \rvert \leqq$ 9999999999	x is an integer.
$x, y \rightarrow r, \theta$		$(x^2 + y^2) < 1 \times 10^{100}$ $\frac{y}{x} < 1 \times 10^{100}$	$r = \sqrt{x^2 + y^2}$ $\theta = \tan^{-1} \frac{y}{x}$
$r, \theta \rightarrow x, y$		$r < 1 \times 10^{100}$, $\lvert r \sin \theta \rvert < 1 \times 10^{100}$ $\lvert r \cos \theta \rvert < 1 \times 10^{100}$	$x = r \cos \theta$ $y = r \sin \theta$ θ is in the same condition as x of $\sin x$, $\cos x$.
Statistical calculation	DATA CD	$\lvert x \rvert < 1 \times 10^{50}$ $\lvert y \rvert < 1 \times 10^{50}$ $\lvert \Sigma x \rvert < 1 \times 10^{100}$ $\Sigma x^2 < 1 \times 10^{100}$ $\lvert \Sigma y \rvert < 1 \times 10^{100}$ $\Sigma y^2 < 1 \times 10^{100}$ $\lvert \Sigma xy \rvert < 1 \times 10^{100}$ $\lvert n \rvert < 1 \times 10^{100}$	
	$\bar{x}$	$n \neq 0$	
	Sx	$n \neq 1$ $0 \leqq \frac{\Sigma x^2 - n\bar{x}^2}{n-1} < 1 \times 10^{100}$	
	σx	$n \neq 0$ $0 \leqq \frac{\Sigma x^2 - n\bar{x}^2}{n} < 1 \times 10^{100}$	
	$\bar{y}$	$n \neq 0$	
	Sy	$n \neq 1$ $0 \leqq \frac{\Sigma y^2 - n\bar{y}^2}{n-1} < 1 \times 10^{100}$	
	σy	$n \neq 0$ $0 \leqq \frac{\Sigma y^2 - n\bar{y}^2}{n} < 1 \times 10^{100}$	

Functions		Dynamic range	Note
Statistical calculation	r	$n \neq 0$ $0 < \lvert (\Sigma x^2 - n\bar{x}^2) \cdot (\Sigma y^2 - n\bar{y}^2) \rvert < 1 \times 10^{100}$ $\left\lvert \Sigma xy - \frac{\Sigma x \cdot \Sigma y}{n} \right\rvert < 1 \times 10^{100}$ $\left\lvert \frac{\Sigma xy - \frac{\Sigma x \cdot \Sigma y}{n}}{\sqrt{(\Sigma x^2 - n\bar{x}^2) \cdot (\Sigma y^2 - n\bar{y}^2)}} \right\rvert < 1 \times 10^{100}$	
	b	$n \neq 0$ $0 < \lvert \Sigma x^2 - n\bar{x}^2 \rvert < 1 \times 10^{100}$ $\left\lvert \Sigma xy - \frac{\Sigma x \cdot \Sigma y}{n} \right\rvert < 1 \times 10^{100}$ $\left\lvert \frac{\Sigma xy - \frac{\Sigma x \cdot \Sigma y}{n}}{\Sigma x^2 - n\bar{x}^2} \right\rvert < 1 \times 10^{100}$	
	a	a is the same condition as b, and $\lvert \bar{y} - b\bar{x} \rvert < 1 \times 10^{100}$	
	y'	$\lvert a + bx \rvert < 1 \times 10^{100}$	
	x'	$\left\lvert \frac{y - a}{b} \right\rvert < 1 \times 10^{100}$	

For the accuracy of functions other than shown above, the error is ±1 at the 10th digit, as a rule. (In the scientific notation system, the error is ±1 at the lowest digit of mantissa display.)
However, the accuracy will become low around singular points and inflection points of functions.

Therefore, errors are accumulated in each stage of the continuous calculations, causing the accuracy to deteriorate. (The same applies to other continuous calculations made by the computer such as y^x and $\sqrt[x]{y}$.)

Matrix Calculation Function

In the CAL mode, the COMPUTER has a function to calculate matrixes or their determinant values.

A matrix is a rectangular array a_{ik} (i = 1, 2 ..., m, k = 1, 2 ..., n) of a given set of numbers (m×n elements) as shown below.

$$\begin{bmatrix} a_{11} & a_{12} \cdots\cdots a_{1n} \\ a_{21} & a_{22} \cdots\cdots a_{2n} \\ \vdots & \\ a_{m1} & a_{m2} \cdots\cdots a_{mn} \end{bmatrix}$$

With this computer, such an array is expressed as matrix X, Y, or M. One of the sets of numbers which form a matrix is called a matrix element. Matrix element a_{11} is expressed as $X(1,1)$, $Y(1,1)$, or $M(1,1)$. The horizontal arrangement of matrix elements is called a row while the vertical arrangement is called a column.

Matrix Configuration

With the COMPUTER, three matrixes X, Y, and M can be defined. Each matrix can be defined within a range of 1 to 99 both vertically (i.e., columns) and horizontally (i.e., rows). However, the total matrix size is dependent upon the memory capacity of the COMPUTER.

In addition, matrixes X, Y, and M are stored in the same memory area as BASIC arrays X(*,*), Y(*,*), and M(*,*). In other words, the values of the matrix elements entered in the BASIC mode can be calculated in the CAL mode.

When entering the values of matrix elements in the BASIC mode, pay attention to the following points:

(1) Matrix elements X(i,k) correspond to BASIC array elements X(i−1, k−1). For example, $X(1,2)$ correspond to array X(0,1).

(2) All the matrix element values stored in memory will be cleared by BASIC command RUN, CLEAR, or NEW.

Input of Matrix Element

In the CAL mode, pressing [SHIFT] [↓] or [SHIFT] [↑] causes the COMPUTER to enter the MATRIX mode. In this mode, you can enter the elements of a matrix for calculation of the matrix, as well as to have the computer perform matrix operations and display the matrix elements entered.

The keys and their functions used to enter and display matrix elements are as described below.

Key	Function
SHIFT ↓	• Puts the computer in the MATRIX mode. • Allows you to enter the elements of matrix X and then the elements of matrix Y, and the computer to calculate the matrixes. • Releases the computer from the MATRIX mode when these keys are pressed a second time.
SHIFT ↑	• Puts the computer in the MATRIX mode. • Allows the computer to perform matrix calculations. • Releases the computer from the MATRIX mode when these keys are pressed a second time.
ENTER	• Stores in memory the number of rows and number of columns which form a matrix and other matrix element data, and then the computer waits for the next data entry.
→	• Shifts the cursor to the right by one column. (When the cursor is at the rightmost column, the cursor moves to the next element.)
←	• Shifts the cursor to the left by one column. (When the cursor is at the leftmost column, the cursor moves to the preceding element.)
↑	• Shifts the cursor up by one row (i.e., to the element immediately above the current column). • Returns the computer to the previous step in operation.
↓	• Shifts the cursor down by one row (i.e., to the element immediately below the current column). • Puts the computer in the wait state for next step in operation.

When you input the respective elements of a matrix, you may use any of the keys that you use in the CAL mode for four basic operations and scientific calculations.

Example 1: To enter the following two matrixes:

$$X = \begin{bmatrix} 10/3 & -5 & 2 \\ 8 & 2 & 23 \end{bmatrix}$$

$$Y = \begin{bmatrix} 5/3 & 3 & 2 \\ -1 & 0 & -8 \end{bmatrix}$$

Operation:

SHIFT ↓	MATRIX:X(Ø _ , Ø)

Because matrix X is undefined, (0, 0) is displayed when the computer is put in the MATRIX mode.

2	MATRIX:X(2 _ , Ø)

"2" is entered as the number of rows.

ENTER	MATRIX:X(2, Ø _)
3 ENTER	X(1,1) Ø.

Then enter the number of columns as "3" and define matrix X as a matrix with a size of (2, 3), and the computer is ready for your input of the value of element $X(1, 1)$.

10 ÷	X(1,1) 1Ø.
3 =	X(1,1) 3.333333333
ENTER	X(1,2) Ø.

After your input of element $X(1, 1)$, the computer waits for your input of the next element $X(1, 2)$.

5 +/−	X(1,2) −5
ENTER 2	X(1,3) 2.
ENTER 8	X(2,1) 8.
ENTER 2	X(2,2) 2.
ENTER 23	X(2,3) 23.
ENTER	MATRIX:Y(Ø _ , Ø)

After you have completed the input of all the element data of matrix X, you must define the size of matrix Y and then enter the elements of matrix Y in the same manner as you did for matrix X.

2 [ENTER]	MATRIX:Y(2, Ø _)
3	MATRIX:Y(2, 3 _)
[ENTER] 5 [÷] 3 [=]	Y(1, 1) 1.666666667
[ENTER] 3	Y(1, 2) 3.
[ENTER] 2	Y(1, 3) 2.
[ENTER] 1 [+/−]	Y(2, 1) −1.
[ENTER] 0	Y(2, 2) Ø.
[ENTER] 8 [+/−]	Y(2, 3) −8.
[ENTER]	MATRIX OPERATION

On input of all the elements of matrix Y, the message "MATRIX OPERATION" will appear on the display, indicating that the COMPUTER is ready to perform matrix calculations. If only matrix X needs to be calculated, press [C·CE] [ENTER] when you input the number of rows and number of columns, respectively, for matrix Y.

Matrix Calculations

While the message "MATRIX OPERATION" is on the display, pressing each of the following keys causes the COMPUTER to perform the matrix operation designated by the key.

Key	Function
[+]	$X + Y \rightarrow X$: Performs addition. The result of adding matrix X to matrix Y becomes new matrix X. To perform addition, matrixes X and Y must be equal to each other in both the number of rows and the number of columns.
[−]	$X - Y \rightarrow X$: Performs subtraction. The result of subtracting some elements of matrix Y from the corresponding elements of matrix X becomes the corresponding elements of new matrix X. To perform subtraction, matrixes X and Y must be equal in both the number of rows and the number of columns. (Example) $\begin{bmatrix} 2 & 3 \\ 5 & 2 \end{bmatrix} - \begin{bmatrix} 1 & 6 \\ 3 & -1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & -3 \\ 2 & 3 \end{bmatrix}$
[×]	$X \cdot Y \rightarrow X$: Performs multiplication. To perform multiplication, the number of columns in matrix X must be equal to the number of rows in matrix Y.
[÷]	$X \cdot Y^{-1} \rightarrow X$: Performs the multiplication of matrix X and inverse matrix Y. To perform this operation, the number of columns in matrix X must be equal to the number of rows in matrix Y^{-1}.
[1/x]	$X^{-1} \rightarrow X$: Performs the inverse matrix calculation of matrix X. The result of this operation becomes new matrix X. To perform this operation, matrix X must be a square matrix (which has the same number of rows as the number of columns).
n [+]	$n + X \rightarrow X$: Performs the addition of scalar n to matrix X elements. In this operation, n is added to each element of matrix X. **NOTE:** Mathematically, such an operation as this does not exist. The addition of scalars is one of the features unique to the COMPUTER.

Key	Function
n [−]	n − $X \rightarrow X$: Performs the subtraction of matrix X elements from scalar n. In this operation, each element of matrix X is subtracted from n and the result becomes the corresponding elements of new matrix X. **NOTE:** Mathematically, such an operation as this does not exist. The subtraction of scalars is another feature unique to the COMPUTER. (Example) $2-\begin{bmatrix}1 & 6\\ 3 & -1\end{bmatrix}\rightarrow\begin{bmatrix}1 & -4\\ -1 & 3\end{bmatrix}$
n [×]	n · $X \rightarrow X$: Performs the multiplication of matrix X elements by scalar n.
n [÷]	n · $X^{-1} \rightarrow X$: Performs the multiplication of inverse matrix X^{-1} elements by scalar n. To perform this operation, matrix X must be a square matrix.
[↕]	$X \longleftrightarrow Y$: Exchange matrix X for matrix Y.
[T]	$X^t \rightarrow X$: Performs the transposition of matrix X, giving the transposed matrix as new matrix X.
[D]	$\lvert X\rvert \rightarrow X$ (Display): Displays the value of the determinant of matrix X. To perform this operation, matrix X must be a square matrix.
[+/−]	$-X \rightarrow X$: Reverses the positive or negative sign of each element of matrix X.
[x^2]	$X \cdot X \rightarrow X$: Perform the squaring of matrix X. To perform this operation, matrix X must be a square matrix.
[X→M]	$X \rightarrow M$: Stores the value of matrix X in the memory location of matrix M (while clearing the previous contents of matrix M). This key is used when you wish to retain the value of matrix X even after the matrix calculation.

Key	Function
[RM]	$M \rightarrow X$: Invokes the memory contents of matrix M into matrix X (while clearing the previous contents of matrix X.)
[M+]	$X + M \rightarrow M$: Adds the value of matrix X cumulatively to the memory contents of matrix M. To perform this operation, matrixes X and M must be equal to each other in both the number of rows and the number of columns.

Note:
- Pressing the [BRK] key during the execution of a matrix calculation causes the calculation to be suspended. At this point, the values of matrixes X, Y, and M will be retained as those before the execution of the calculation.
- Press the n [1/x] [X] in this order to perform the division of matrix X elements by scalar n.
- If most of the elements of a matrix have the same value, execute the n [+] operation with all the matrix elements set as 0 and then correct only the value of each element having a value other than n. This will facilitate the input of the matrix elements.

On completion of the matrix calculation, the message "MATRIX OPERATION" appears again on the display, indicating that the COMPUTER is ready for the next matrix calculation. After the determinant value of matrix X is displayed by pressing the [D] key, the message "MATRIX OPERATION" will appear again if you press one of the [↓], [↑], [←], [→], [C·CE], and [ENTER] keys.

Example 2: To calculate $X + Y$, using the values of the respective elements of matrixes X and Y entered (stored in memory) in Example 1

$$X = \begin{bmatrix} 10/3 & -5 & 2 \\ 8 & 2 & 23 \end{bmatrix} \qquad Y = \begin{bmatrix} 5/3 & 3 & 2 \\ -1 & 0 & -8 \end{bmatrix}$$

Operation:

[+]

MATRIX OPERATION

X+Y→X

(The message "BUSY" appears indicating that the computer is performing a calculation.)

MATRIX OPERATION

Now, let's see the result of addition.

[↓]	MATRIX:X(2 _ , 3)
[↓]	X(1, 1) 5.
[►]	X(1, 2) −2.
[SHIFT] [↓]	Ø.

The COMPUTER is now released from the MATRIX mode.

If you press any of the numeric keys or the [•] key while the message "MATRIX OPERATION" is being displayed, the COMPUTER can perform scalar calculations.

Example 3: To calculate $1/25 * X \rightarrow X$, using the calculation result of matrix X in Example 2

Operation:

[SHIFT] [↑]	MATRIX OPERATION
2	SCALAR 2.
5	SCALAR 25.
[1/x]	SCALAR Ø.Ø4
[×]	Ø.Ø4*X→X
	MATRIX OPERATION
[↓]	MATRIX:X(2_, 3)
[↓]	X(1, 1) Ø.2
[SHIFT] [↑]	Ø.

Example 4: To solve the following simultaneous linear equations with three unknowns using matrix calculations

$$\begin{cases} 2x + 5y - z = -1 \\ x - y + 4z = 12 \\ 3x + 2y + z = 9 \end{cases}$$

HINTS: Enter matrixes X and Y as shown below and calculate $X^{-1} \times Y$ to obtain the solutions x, y, and z of the equations.

$$X = \begin{bmatrix} 2 & 5 & -1 \\ 1 & -1 & 4 \\ 3 & 2 & 1 \end{bmatrix} \qquad Y = \begin{bmatrix} -1 \\ 12 \\ 9 \end{bmatrix}$$

Operation:

Press the [SHIFT] and [↓] keys to put the computer in the MATRIX mode and then enter the matrix element data of X and Y according to Example 1.

	MATRIX OPERATION
[1/x]	invX→X
	MATRIX OPERATION
[×]	X*Y→X
	MATRIX OPERATION
[↓]	MATRIX:X(3 _ , 1)
[↓]	X(1, 1) 3.
[↓]	X(2, 1) −1.
[↓]	X(3, 1) 2.

Thus, the solutions x, y, and z of the equations are as follows:

x = 3, y = −1, z = 2

Note: Matrix calculations are based on the method of elimination being widely used. However, due to the nature of numerical calculations by any computer, an error may occur in the calculation of a determinant or an inverse matrix because of truncation or some other reasons.

Example 5: To solve for the inverse matrix of $\begin{bmatrix} 3 & 1 \\ 1 & 1/3 \end{bmatrix}$

This matrix is not a regular matrix and thus has no inverse matrix theoretically. With any computer, however, the value 1/3 is input as "0.33 .....3" and thus an inverse matrix exists, resulting in the following.

$$\begin{bmatrix} 3 & 1 \\ 1 & 0.33\cdots3 \end{bmatrix}^{-1} = \begin{bmatrix} -33\cdots3. & 1.E10 \\ 1.E10 & -3.E10 \end{bmatrix}$$

So the results obtained by computers may have such an error. Please note that verification by any other method may be required depending on how matrix calculations will be applied.

In the above example, when you obtain the determinant value by multiplying the original matrix X by 3, you can confirm that matrix X is not a regular matrix because the result of the multiplicaiton becomes 0 ($\begin{vmatrix} 9 & 3 \\ 3 & 1 \end{vmatrix} = 0$).

Note: Because a matrix calculation will not be completed by a single operation (e.g., one-time multiplication), it will take some time to complete the calculation. It will take about 6 seconds to solve for the inverse matrix of a unit matrix consisting of 7 rows and 7 columns. This calculation time varies depending on the values of matrix elements.

Memory Capacity Required for Matrix Calculations

- Because matrix calculations share the same memory area as that used for BASIC programs, unused memory capacity (i.e., capacity determinable by MEM [ENTER] in BASIC mode) must be larger than the capacity determined by the following formula:

[(No. of rows of matrix X) × (No. of columns of matrix X) × 8 + 7] bytes
+ [(No. of rows of matrix Y) × (No. of columns of matrix Y) × 8 + 7] bytes
+ [No. of rows of matrix M) × (No. of columns of matrix M) × 8 + 7] bytes
+ [No. of rows of resultant matrix) × (No. of columns of resultant matrix) × 8 + 7] bytes

However, when neither matrix Y nor matrix M is used, the values (no. of rows and no. of columns) in brackets of each unused matrix will be treated as 0 for the capacity calculation. The resultant matrix is only required during the calculation and will be cleared on completion of the calculation. For information, no resultant matrix is required for the execution of [x→M], [RM], or [↕]. Two resultant matrixes are required for matrix operations using [÷] and n [÷], since these operations involve two calculations (i.e., inversion and multiplication).

- If the message "MEMORY OVER" appears on the display while in the MATRIX mode, erase the variables or programs used in BASIC in order to increase the unused memory capacity for matrix calculations.
  When calculating matrixes consisting of numerous elements, use a RAM card having a larger capacity.

Example 6: To calculate the multiplication of the following two matrixes ($X \cdot Y \rightarrow X$)

$$X = \begin{bmatrix} 2 & 3 \\ 5 & 1 \end{bmatrix} \quad Y = \begin{bmatrix} 8 & 30 \\ 7 & 15 \end{bmatrix}$$

(matrix M undefined)

The required memory capacity will be calculated as follows:

$$[2 \times 2 \times 8 + 7] + [2 \times 2 \times 8 + 7] + [2 \times 2 \times 8 + 7] = 117 \text{ bytes}$$

Matrix X | Matrix Y | Resultant matrix

Printing of Matrixes

To print the data (e.g., value of each element) of matrix X, prepare and execute the following program. If you execute the program by typing "RUN" and pressing [ENTER], however, all the matrix data will be cleared from memory. So be sure to execute the program with the [DEF] key.

```
100 "M":INPUT "ROW";II
110 INPUT "COLUMN";JJ
120 FOR I=0 TO II-1
130 FOR J=0 TO JJ-1
140 LPRINT "X(";I+1;",";J+1;")=";X(I,J)
150 NEXT J:NEXT I:END
```

If the data of matrix Y or M is to be printed, change "X" at the two places in line 140 of the above program to read "Y" or "M".

(Operation) Press the [DEF] [M] keys in the RUN mode, and the designated matrix data will be output on the printer.

Error Messages

If an error occurs during the calculation of a matrix, one of the following messages appears on the display, together with the "E" (Error) sign. Press the [C·CE] key to release the COMPUTER from the error condition and the "E" sign will go off and the message "MATRIX OPERATION" will appear on the display. At this point, the matrix data before the execution of the matrix calculation is retained in memory.

Error Message	Cause of Error
IMPOSSIBLE CALCULATION	• Matrixes do not match in size. Matrixes do not match in size in addition, subtraction, or multiplication, or an attempt was made to calculate the inverse matrix, or to perform the squaring, of a matrix which is not a square matrix.
MEMORY OVER	• Insufficient memory In $X \rightarrow M$ operation, memory space is not enough to store matrix M, or no work area is available for arithmetic operation.
DIVISION BY ZERO	• 0 (zero) is used as divisor. In an inverse matrix calculation, an attempt was made to divide a number by zero.
OVER FLOW	• Overflow has occurred during an arithmetic operation.

Manual Calculations in BASIC Mode

What is Manual Calculation?

The **COMPUTER** may be basically used in two ways. One way lets you store in advance the whole calculation procedure or steps into the computer's memory as a program, then lets the computer automatically execute it later. The other lets you calculate step by step through manual key operations. The latter is called a manual calculation.

Of course, in the CAL mode, all calculations are performed manually, but here only those performed in the BASIC mode (RUN or PROgram mode) are called manual calculations.

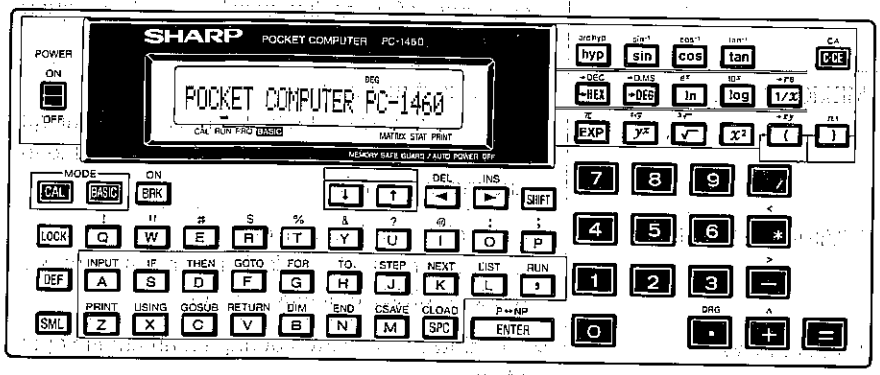
How to Manually Calculate

Let's try manual calculation in the RUN mode. Press the BASIC key to place your computer in the RUN mode.

```
RUN
```

Press [BASIC]; an indicator will appear with the RUN label.

In the RUN mode, the key functions shown in the following figure are operative. (The same is true in the PROgram mode.)



Before going into operation examples, let's touch on some important points in operation.

Whereas we usually use operators +, −, ×, or ÷ for our mathematical calculations on paper, we don't use the operators × and ÷ for our arithmetic operations in BASIC. Instead of × and ÷, we use an asterisk (*) and slash (/), respectively.

The operators * and / can be entered by pressing [×*] and [÷/] keys, respectively. **To get the result of a manual calculation, operate the [ENTER] key instead of [=] key.**

Do not use dollar signs or commas when entering a calculation formula or expression into the COMPUTER. These characters have special meaning in the BASIC programming language. Now try these simple arithmetic examples. Remember to clear with the [C·CE] between calculations.

Input	Display
5 Ø + 5 Ø ENTER	1ØØ.
1 Ø Ø − 5 Ø ENTER	5Ø.
6 Ø * 1 Ø ENTER	6ØØ.
3 Ø Ø / 5 ENTER	6Ø.
1 Ø SHIFT ^ 2 ENTER	1ØØ.
2 * SHIFT π ENTER	6.283185307
√ 6 4 ENTER	8.

Recalling Entries

Even after the **COMPUTER** has displayed the results of your calculation, you can display your last entry again. To recall, use the left ◄ and right ► arrows.

The left arrow, ◄, recalls the expression that has the cursor positioned after its last character.

The right arrow, ►, recalls the expression that has the cursor positioned "on top of" its first character.

Remember that the left and right arrows are also used to position the cursor within a line. The right and left arrows are very helpful in editing (or modifying) entries without having to retype the entire expression.

You will become familiar with the use of the right and left arrows in the following examples. Now, take the role of the manager and perform the calculations as we discuss them.

As the head of personnel in a large marketing division, you are responsible for planning the annual sales meeting. You expect 300 people to attend the three-day conference. For part of this time, the sales force will meet in small groups. You believe that groups of six would be a good size. How many groups would this be?

Input	Display
3 Ø Ø / 6 ENTER	5Ø.

On second thought, you decide that groups containing an odd number of participants might be more effective. Recall your last entry, using the ◄ key

Input	Display
◄	300/6_

To calculate the new number of groups, you must replace the six with an odd number. Five seems to make more sense than seven. Because you recalled by using the ◄ arrow, the cursor is positioned at the end of the display. Use the ◄ to move the cursor one space to the left.

Input	Display
◄	300/6

Notice that after you move the cursor it becomes a flashing block ▓. Whenever you position the cursor "on top of" an existing character, it will be displayed as a flashing cursor.

Type in a 5 to replace the 6. One caution in replacing characters–once you type a new character over an existing character, the original is gone forever! You cannot recall an expression that has been typed over.

Input	Display
5	300/5_
ENTER	60.

Sixty seems like a reasonable number of groups, so you decide that each small group will consist of five participants.

Recall is also useful to verify your last entry, especially when your results do not seem to make sense. For instance, suppose you had performed this calculation:

Input	Display
3 0 / 5 ENTER	6.

Even a tired, overworked manager like you realizes that six does not seem to be a reasonable result when you are dealing with hundreds of people! Recall your entry using the [►] .

Input	Display
[►]	30/5

Because you recalled by using the [►] , the flashing cursor is now positioned over the first character in the display. To correct this entry you wish to insert an added zero. Using the [►] , move the cursor until it is positioned over the zero. When making an INSert, you position the flashing cursor over the character **before** which you wish to make the insertion.

Input	Display
[►]	30/5

Use the INSert key to make space for the needed character.

Input	Display
[SHIFT] [INS]	3□0/5

Pressing INSert moves all the characters one space to the right, and inserts a bracketed open slot. The flashing cursor is now positioned over this open space, indicating the location of the next typed input. Type in your zero. Once the entry is corrected, display your new result.

Input	Display
[0]	300/5
[ENTER]	60.

On the other hand, suppose that you had entered this calculation:

Input	Display
[3] [0] [0] [0] [/] [5] [ENTER]	600.

The results seem much too large. If you only have 300 people attending the meeting, how could you have 600 "small groups"? Recall your entry using the [►] .

Input	Display
[►]	3Ø0/5

The flashing cursor is now positioned over the first character in the display. To correct this entry eliminate one of the zeros. Using the [►] , move the cursor to the first zero (or any zero). When deleting a character, you position the cursor "on top of" the character to be deleted.

Input	Display
[►]	3Ø0/5

Now use the DELete key to get rid of one of the zeros.

Input	Display
[SHIFT] [DEL]	3Ø/5

Pressing DELete causes all the characters to shift one space to the left. It deletes the character it is "on top of" and the space the character occupied. The flashing cursor stays in the same position indicating the next location for input. Since you have no other changes to make, complete the calculation.

Input	Display
[ENTER]	6Ø.

Note: Pressing the SPaCe key, when it is positioned over a character, replaces the character leaving a blank space. DELete eliminates the character and the space it occupied.

Errors

Recalling your last entry is essential when you get the dreaded ERROR message. Let us imagine that, unintentionally, you typed this entry into the computer.

Input	Display
[3] [Ø] [Ø] [/] [/] [5] [ENTER]	ERROR 1

Naturally you are surprised when this message appears! ERROR 1 is simply the computer's way of saying, "I don't know what you want me to do here". To find out what the problem is, recall your entry using either the ◄ or ► arrow.

Input	Display
◄	300//5

When you use the ◄ or ► key, the flashing cursor indicates the point at which the computer got confused. And no wonder, you have too many operators! To correct this error use the DELete key.

Input	Display
SHIFT DEL ENTER	60.

If, upon recalling your entry after an ERROR 1, you find that you have omitted a character, use the INSert sequence to correct it.

When using the computer as a calculator, the majority of the errors you encounter will be ERROR 1 (an error in syntax). For a complete listing of error messages, see Appendix A.

Serial Calculations

The computer allows you to use the results of one calculation as part of the following calculation.

Part of your responsibility in planning this conference is to draw up a detailed budget for approval. You know that your total budget is $150.00 for each attendant. Figure your total budget:

Input	Display
3 0 0 * 1 5 0 ENTER	45000.

Of this amount you plan to use 15% for the final night's awards presentation. When performing serial calculations, it is not necessary to retype your previous results, but DO NOT clear between entries (do not use the C·CE at this time). What is the awards budget?

Input	Display
* . 1 5	45000.*.15_

Notice that as you type in the second calculation (*.15), the computer automatically displays the result of your first calculation at the left of the screen and includes it in the new calculation. In serial calculations, the entry must begin with an operator. As always, you end the entry with [ENTER] :

Note: The [%] and [Δ% D] keys cannot be used in the calculation. The [%] key should be used as a character only and the [Δ% D] key is inoperative.
Example: 45000 [*] 15 [SHIFT] [%] → ERROR1

Input	Display
[ENTER]	6750.

Continue allocating your budget. The hotel will cater you dinner for $4000:

Input	Display
[−] [4] [0] [0] [0]	6750.−4000_
[ENTER]	2750.

Decorations will be $1225:

Input	Display
[−] [1] [2] [2] [5] [ENTER]	1525.

Finally, you must allocate $2200 for the guest speaker and entertainment:

Input	Display
[−] [2] [2] [0] [0] [ENTER]	−675.

Obviously, you will have to change either your plans or your allocation of resources!

Negative Numbers

Since you want the awards dinner to be really special, you decide to stay with the planned agenda and spend the additional money. However, you wonder what percentage of the total budget will be used up by this item. First, change the sign of the remaining sum:

Input	Display
[*] [-] [1]	-675.*-1_
[ENTER]	675.

Now you add this result to your original presentation budget:

Input	Display
[+] [6] [7] [5] [Ø] [ENTER]	7425.

Dividing by 45000 gives you the percentage of the total budget this new figure represents:

Input	Display
[/] [4] [5] [Ø] [Ø] [Ø] [ENTER]	Ø.165

Fine, you decide to allocate 16.5% to the awards presentation.

Compound Calculations and Parentheses

In performing the above calculations, you could have combined several of these operations into one step. For instance, you might have typed both these operations on one line:

675+6750/45000

Compound calculations, however, must be entered very carefully:

675+6750/45000 might be interpreted as

$$\frac{675+6750}{45000} \quad \text{or} \quad 675+\frac{6750}{45000}$$

When performing compound calculations, the computer has specific rules of expression evaluation and operator priority (see page 78). Be sure you get the calculation you want by using parentheses to clarify your expressions.

(675+6750)/45000 or 675+(6750/45000)

To illustrate the difference that the placement of parentheses can make, try these two examples:

Input	Display
[(] [6] [7] [5] [+] [6] [7] [5] [Ø] [)] [/] [4] [5] [Ø] [Ø] [Ø] [ENTER]	Ø.165
[6] [7] [5] [+] [(] [6] [7] [5] [Ø] [/] [4] [5] [Ø] [Ø] [Ø] [)] [ENTER]	675.15

Note: In BASIC (PRO or RUN) mode, the close parenthesis before the [ENTER] key cannot be omitted. In CAL mode, however, you can omit it before the [=] key.

Using Variables in Calculations

The computer can store up to 26 fixed variables under the alphabetic characters A to Z. If you are unfamiliar with the concept of variables, they are more fully explained in Chapter 4. You designate variables with an Assignment Statement:

```
A=5
B=-2
```

You can also assign the value of one variable (right) to another variable (left):

```
C = A + 3
D = E
```

A variable may be used in place of a number in any calculation.

Now that you have planned your awards dinner, you need to complete arrangements for your conference. You wish to allocate the rest of your budget by percentages also. First you must find out how much money is still available. Assign a variable (R) to be the amount remaining from the total:

Input	Display
[R] [=] [4] [5] [Ø] [Ø] [Ø] [−] [7] [4] [2] [5]	R=45ØØØ−7425_
[ENTER]	37575.

As you press [ENTER], the computer performs the calculation and displays the new value of R. You can display the current value of any variable by entering the alphabetic character it is stored under:

Input	Display
[R] [ENTER]	37575.

You can then perform calculations using your variable. The value of (R) will not change until you assign it a new value.

You wish to allocate 60% of the remaining money to room rental:

Input	Display
[R] [*] [.] [6] [Ø]	R*.6Ø_
[ENTER]	22545.

Similarly, you want allocate 25% of your remaining budget to conduct management training seminars:

Input	Display
[R] [*] [.] [2] [5] [ENTER]	9393.75

Variables will retain their assigned values even if the machine is turned OFF or undergoes an AUTO OFF. Variables are lost only when you:

* assign a new value to the same variable.
* type in CLEAR [ENTER] (not the clear key ([C·CE])).
* clear the machine using the RESET button.
* change the batteries.

These are certain limitations on the assignment of variables, and certain programming procedures which cause them to be changed. See Chapter 4 for a discussion of assignment. See Chapter 5 for a discussion of the use of variables in programming.

Chained Calculations

In addition to combining several operators in one calculation, the computer also allows you to perform several calculations one after another–without having to press [ENTER] before moving on. You must separate the equations with commas. Only the result of the **final** calculation is displayed. (Remember too that the maximum line length accepted by the computer is 80 characters including [ENTER] .)

You wonder how much money would have been available for rooms if you had kept to your original allocation of 15% for the awards dinner:

Input	Display
[R] [=] [.] [8] [5] [*] [4] [5] [Ø] [Ø] [Ø] [,] [R] [*] [.] [6] [Ø]	R=.85*45ØØØ,R*.6Ø_

Although the computer performs all the calculations in the chain, it displays only the final result:

Input	Display
[ENTER]	22950.

To find the value of R used in this calculation, enter R:

Input	Display
[R] [ENTER]	38250.

Error Message

If an error occurred as a result of a manual calculation, an error message will appear in the display such as:

ERROR 1 or ERROR 2

The error state can be cleared with either the [C-CE] or [◄] or [►] key. If the [◄] or [►] key is used to clear the error state, the portion of the formula where the error occurred is recalled in the display (see the description for the recall feature).

Scientific Notation

People who need to deal with very large and very small numbers often use a special exponential format called **scientific notation.** In scientific notation, a number is broken down into two parts.

The first part (called mantissa part) consists of a regular decimal number between 1 and 10. The second part (called exponent part) represents how large or small the number is in powers of 10.

As you know, the first number to the left of the decimal point in a regular decimal number shows the number of 1's, the second shows the number of 10's, the third the number of 100's, and the fourth the number of 1000's. These are simply increasing powers of 10:

$10^0 = 1$, $10^1 = 10$, $10^2 = 100$, $10^3 = 1000$, etc.

Scientific notation breaks down a decimal number into two parts: one shows what the numbers are, the other shows how far a number is to the left, or right, of the decimal point. For example:

1234 becomes 1.234 times 10^3 (3 places to the right)
654321 becomes 6.54321 times 10^5 (5 places to the right)
.000125 becomes 1.25 times 10^{-4} (4 places to the left)

Scientific notation is useful for many short cuts. You can see that it would take a lot of writing to show 1.0 times 10^{87} or 1 and 87 zeros! But, in scientific notation, this number looks like this:

1.0×10^{87} or 1.0E 87

The computer uses scientific notation whenever numbers become too large to display using decimal notation. This computer uses the capital letter **E** to mean "times ten to the":

1234567890000 is displayed as 1.23456789**E** 12
.000000000001 is displayed as 1. **E** −12

Those of you who are unfamiliar with this type of notation should take some time to put in a few very large and very small numbers to note how they are displayed.

Limits

The largest number which the computer can handle is ten significant digits, with two digit exponents. In other words, the largest number is:

9.999999999E 99 = 99999999990000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000

and the smallest number is:

9.999999999E −99 = .000000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000099
99999999

Under certain circumstances, when numbers will be used frequently, the computer uses a special compact form. In these cases, there are special limits imposed on the size of numbers, usually either 0 to 65535 or −32768 to +32767. Numbers within this range can be represented in 16 binary bits. The circumstances in which this form is used are noted in Chapter 8.

Last Answer Feature

In the case of the serial calculation, you could use the result of the calculation only as the first member of the subsequent calculation formula.
Refer to the following example.

Input	Display
3 [+] 4 [ENTER]	7.
[*] 5	7.*5_
[ENTER]	35.

Press [C·CE] , then the [↓] or [↑] key. If you operated these keys just after completing the calculation example above, you should see "35." in your display. The numeric data displayed is the result of the last calculation.

The computer can "remember" the last answer (result) obtained through a manual calculation, and recall it on its display with the [↓] or [↑] key.

In the case of the serial calculation described above, you could use the result of the previous calculation only as the first member of the subsequent calculation formula. With the last answer feature, however, you can place the result of the previous calculation in any position of the subsequent calculation formula.

Example: Use the result (6.25) of the operation, 50 ÷ 8, to compute 12 × 5 ÷ 6.25 + 24 × 3 ÷ 6.25 =:

Input	Display
50 [/] 8 [ENTER]	6.25
	Last answer ↑
12 [*] 5 [/] [↑]	12*5/6.25_
	Last answer recalled
[+] 24 [*] 3 [/] [↓]	12*5/6.25+24*3/6.25_
	Last answer recalled
[ENTER]	21.12
[C-CE] [↓]	21.12_

The last answer is replaced with the result of the previous calculation by performing a manual calculation with the [ENTER] key.

As shown in this example, the last answer can be recalled as many times as required, but will be replaced with a new last answer resulting from the last calculation.

The last answer is not cleared by the [C-CE] or [SHIFT] [CA] key operation.

Note: The last answer cannot be recalled when the program execution is temporarily halted in other than the RUN mode, or when the program is under execution in the TRACE mode.

Length of Formula

The length of a formula you can put into your computer has a certain limitation. With the computer, up to 79 key strokes can be used to enter a single calculation formula (excluding the [ENTER] key). If you attempt the 80th key stroke, the cursor (▒) will start blinking on that character, indicating that the 80th key entry is not valid.

Scientific Calculations in the BASIC mode

This computer has many scientific functions which can be used in BASIC mode.

To perform scientific functions you must press [ENTER] at the end of the input, or your calculations will not be acted upon by the computer.

These functions will be described as follows:

Functions	Notation	Operation	Remark
Trigonometric functions			
sin	SIN	[sin]	
cos	COS	[cos]	
tan	TAN	[tan]	
Inverse trigonometric functions			
$\sin^{-1}$	ASN	[SHIFT] [sin^{-1}]	
$\cos^{-1}$	ACS	[SHIFT] [cos^{-1}]	
$\tan^{-1}$	ATN	[SHIFT] [tan^{-1}]	
Hyperbolic functions			
sinh	ASN	[hyp] [sin]	
cosh	ACS	[hyp] [cos]	
tanh	ATN	[hyp] [tan]	

Functions	Notation	Operation	Remark
Inverse hyperbolic functions			
$\sinh^{-1}$	AHS	SHIFT hyp $\sin^{-1}$	
$\cosh^{-1}$	AHS	SHIFT hyp $\cos^{-1}$	
$\tanh^{-1}$	AHT	SHIFT hyp $\tan^{-1}$	
Logarithmic functions			
ln	LN	ln	$\log_e x$
log	LOG	log	$\log_{10} x$
Exponential functions			
e^x	EXP	SHIFT e^x	$e \fallingdotseq 2.718281828$
10^x	TEN	SHIFT 10^x	
Reciprocal $\frac{1}{x}$	RCP	$1/x$	
Square x^2	SQU	x^2	
Square root $\sqrt{\ }$	$\sqrt{\ }$ or SQR	$\sqrt{\ }$	
Cubic root $\sqrt[3]{\ }$	CUR	SHIFT $\sqrt[3]{\ }$	
Factorial n!	FACT	SHIFT $n!$	
Pi	π or PI	SHIFT π	$\pi \fallingdotseq 3.141592654$
DMS → DEG	DEG	°DEG	
DEG → DMS	DMS	SHIFT °DMS	
Power y^x	$\wedge$	SHIFT $\wedge$ or y^x	$y \wedge x : y^x$
Power root $\sqrt[x]{y}$	ROT	SHIFT $\sqrt[x]{y}$	y ROT $x : \sqrt[x]{y}$
Rectangular coordinates → Polar coordinates	POL	SHIFT $\rightarrow r\theta$	

Functions	Notation	Operation	Remark
Polar coordinates → Rectangular coordinates	REC	SHIFT +xy	
Integer	INT	I N T	INT (x)
Absolute $\lvert x \rvert$	ABS	A B S	ABS (x)
Sign	SGN	S G N	SGN (x) $x>0$: 1 $x=0$: 0 $x<0$: −1
Modify (Rounding)	MDF	M D F	

Of these functions, the INT, ABS, SGN, and MDF can be entered by using letter keys. Some other functions may also be entered with letter keys. For example, "sin 30" may be entered either by operating sin 30 or S I N 30. For trigonometric and inverse trigonometric functions and coordinates conversion, the desired angular unit must be specified in advance. In manual calculations, angular units may be specified either by operating SHIFT DRG as in the CAL mode or with the following commands:

Angular unit	Command	Display Symbol	Description
Degree	DEGREE	DEG	Represents a right angle as 90 [°].
Radian	RADIAN	RAD	Represents a right angle as $\pi/2$ [rad].
Grad	GRAD	GRAD	Represents a right angle as 100 [g].

These commands are used to specify angular units in a program. For practice, use them in the following calculation examples:

Example: sin 30°=

Operation: DEGREE ENTER (Specifies "degrees" for angular unit.)

SIN 30 ENTER

DEG
Ø.5

Example: $\tan \frac{\pi}{4} =$

Operation: RADIAN [ENTER] (Specifies "radians" for angular unit.)
TAN (PI/4) [ENTER]

RAD 1.

Example: $\cos^{-1}(-0.5) =$

Operation: DEGREE [ENTER] (Specifies "degrees" for angular unit.)
ACS−0.5 [ENTER]

DEG 120.

(120°)

Example: log 5 + ln 5 =

Operation: LOG 5 + LN 5 [ENTER]

2.308Ø7917

Example: $e^{2+3} =$

Operation: EXP (2 + 3) [ENTER]
(Do not use the [EXP] key.)

148.4131591

Example: $\sqrt[3]{4^3 + 5^3} =$

Operation: CUR (4 ^ 3 + 5 ^ 3) [ENTER]

5.738793548

Example: Convert 30 deg. 30 min. in sexagenary notation into its decimal equivalent (in degrees).

Operation: DEG 30.30 [ENTER]

3Ø.5

(30.5 degrees)

Example: Convert 30.755 deg. in decimal notation into its sexagenary equivalent (in degrees, minutes, seconds).

Operation: DMS 30.755 [ENTER]

3Ø.4518

(30 deg. 45 min. 18 sec.)

Example: Conversion from rectangular into polar coordinates: Determine polar coordinate (r,θ) for point P (3, 8) on a rectangular coordinate:

Operation: DEGREE [ENTER] (Specifies "degrees" for angular unit.)
POL (3, 8) [ENTER] (r) 8.544Ø03745

Z [ENTER] (θ) 69.44395478

* The value of θ is stored in variable Z, and the value of r in variable Y.

Example: Conversion from polar into rectangular coordinates: Determine rectangular coordinates (x, y) for point P (12, 4/5π) on polar coordinates.

Operation: RADIAN [ENTER] Specifies "radians" for angular unit.)
REC (12, (4/5*PI))

[ENTER] (x) −9.708203933
($x \fallingdotseq -9.7$)

Z [ENTER] (Y) 7.053423028
($y \fallingdotseq 7.1$)

* The values of y and x are stored in variables Z and Y, respectively.

Note: For coordinates conversion, the conversion results are stored in variables Z and Y. Therefore, the previous contents of Z and Y (or Z$ and Y$) will be cleared.

– Reference –

Equations composed of logical operators (=, >, <, >=, <=, <>) can take on the values listed in the following table:
x and y represent numeric values.

=*	1 if $x = y$ 0 if $x \neq y$	>=	1 if $x \geqq y$ 0 if $x < y$
>	1 if $x > y$ 0 if $x \leqq y$	<=	1 if $x \leqq y$ 0 if $x > y$
<	1 if $x < y$ 0 if $x \geqq y$	<>	1 if $x \neq y$ ("<>" means "≠".) 0 if $x = y$

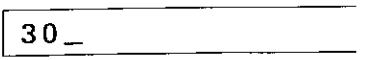
* If, for example, "A = numeric value" or "B = formula" is used in a logical equation, the computer will not treat it as a logical equation but as an assignment statement for variables. When using an equal (=) sign for a logical equation, use it in the form of "numeric value = A" or "formula = B", with the exception of conditional expressions used in IF statements.

Direct Calculation Feature

In the manual calculations described up to now, we always used the [ENTER] key to terminate a formula and obtain the calculation result of the formula. However, you can directly operate the functions of the computer with the desired function key (without operating the [ENTER] key) when the objective numeric data is in the display.

Example: Determine sin 30° and 8!.
Operation: DEGREE [ENTER]

[C·CE] 30



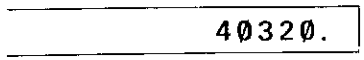
[sin]



Operation: [C·CE] 8

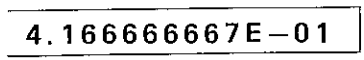


[SHIFT] [n!]



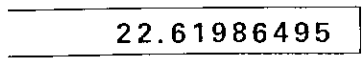
Example: For $\tan^{-1}\frac{5}{12}$, first check the result of $\frac{5}{12}$, then determine $\tan^{-1}\frac{5}{12}$.

Operation: DEGREE [ENTER]



5/12 [ENTER]

[SHIFT] [tan⁻¹]



It should be noted, however, that this "direct" calculation mode is not available for functions requiring the entry of more than one numeric value (binominal functions) such as power, power root, or coordinates conversion.
The direct calculation feature is not effective for formulas:

e.g., [C·CE] 5*4 → 5*4_
[log] → 5*4LOG_

The direct calculation feature is effective only for numeric values. Therefore, if hex numbers A to F are entered for hex to decimal conversion, the direct calculation feature will remain inoperative. In such a case, use the ordinary manual calculation using the [ENTER] key.

Note: After a direct calculation is done, the recall feature is not operative. Operation of the [◄] or [►] key will only display the cursor.

Priority in Manual Calculations

In the BASIC mode, you can type in formulas in the exact order in which they are written, including parentheses or functions. The order of priority in calculation and treatment of intermediate results will be taken care of by the computer itself.

The internal order of priority in manual calculations is as follows:

1) Recalling variables or π.
2) Functions (SIN, COS, etc.)
3) Power (^) and power root (ROT)
4) Signs (+, −)
5) Multiplication and division (*, /)
6) Addition and subtraction (+, −)
7) Comparison of magnitude (>,>=, <, <=, <>)
8) Logical operations (AND, OR, NOT, and XOR)

Notes:
- If parentheses are used in a formula, the operation given within the parentheses has the highest priority.
- Composite functions are operated from right to left (sin $\cos^{-1}$ 0.6).
- Chained power (3^{4^2} or 3^4^2) or power root are operated from right to left.
- For the above items 3) and 4), the last entry in the calculation formula has a higher priority.

(e.g.) −2^4 → $-(2^4)$

3^−2 → 3^{-2}

CHAPTER 4
CONCEPTS AND TERMS OF BASIC

In this chapter, we will examine some concepts and terms of the BASIC language.

String Constants

In addition to numbers, there are many ways that the **SHARP COMPUTER** uses letters and special symbols. These letters, numbers, and special symbols are called characters. These characters are available on the computer.

1 2 3 4 5 6 7 8 9 Ø
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
! " # $ % & ( ) * + , – . / : ; < = > ? @ √ π ^

In BASIC, a collection of characters is called a **string**. For the computer to tell the difference between a string and other parts of a program, such as verbs or variable name, you must enclose the characters of the string in quotation marks (").
The following are examples of string constants:

"HELLO"
"GOODBYE"
"SHARP COMPUTER"

The following are not valid string constants:

"COMPUTER	No ending quote
"ISN"T"	Quote can't be used within a string

Hexadecimal Numbers

The decimal system is only one of many different systems to represent numbers. Another which has become quite important when using computers is the hexadecimal numbering system. The hexadecimal system is based on 16 instead of 10. To write hexadecimal numbers, you use the familiar 0 to 9 and 6 more "digits": A, B, C, D, E, and F. These correspond to 10, 11, 12, 13, 14, and 15. When you want the computer to treat a number as hexadecimal, put an ampersand '&' character in front of the numeral:

&A = 1Ø
&1Ø = 16
&1ØØ = 256
&FFFF = 65535

Those with some computer background may notice that the last number (65535) is the same as the largest number in the special group of limits discussed in the paragraph "Limits" on page 69. Hexadecimal notation is never absolutely necessary in using the computer, but there are special applications where it is convenient.

Variables

Computers are made up of many tiny memory areas called bytes. Each byte can be thought of as a single character. For instance, the word byte requires four bytes of memory because there are four characters in it. To see how many bytes are available for use, simply type in MEM [ENTER]. The number displayed is the number of bytes available for writing programs. This technique works fine for words, but is very inefficient when you try to store numbers. For this reason, numbers are stored in a coded fashion. Thanks to this coding technique, your computer can store large numbers in only eight bytes. The largest number that can be stored is +9.999999999 E + 99.

The smallest number is +1.E−99. This gives you quite a range to choose from. However, if the result of a calculation exceeds this range, the computer will let you know by turning on the error annunciator and by displaying the error message in the screen. This annunciator is a small E in the upper right-hand corner of the screen. For the error message, refer to Appendix A. To see it right now type in:

9 [EXP] 99 * 9 [ENTER]

To get the computer working properly again, just press the [C·CE] key. But how do you go about storing all this information? It's really very easy. The computer likes to use names for different pieces of data. Let's store the number 556 into the computer. You may call this number by any name that you wish, but for this exercise, let's use the letter R. The statement, LET, can be used to instruct the computer to assign a value to a variable name but only in a program statement. However, the LET command is not necessary, so we will not use it very often. Now, type in R = 556 and press the [ENTER]. The computer now has the value 556 associated with the letter R. These letters that are used to store information are called **variables**. To see the content of the variable R, press the [C·CE] key, the letter R key, and the [ENTER] key. The computer responds by showing you the value 556 on the right of your screen. This ability can become very useful when you are writing programs and formulas.

Next, let's use the R variable in a simple formula. In this formula, the variable R stands for the radius of a circle whose area we want to find. The formula for the area of a circle is: $A = \pi * R^2$. Type in R [SHIFT] [^] 2 [*] [SHIFT] [π] [ENTER]. The result is 971179.3866. This technique of using variables in equations will become more understandable as we get into writing programs.

So far, we've only discussed numeric variables. What about storing alphabetic characters? Well, the idea is the same, but, so the computer will know the difference between the two kinds of variables, add a $ to the variable name. For instance, let's store the word BYTE in the variable B$. Notice the $ after the B?
This tells the computer that the contents of the letter B are alphabetic, or string data.

To illustrate this, key in B [SHIFT] [$ R] = [SHIFT] [" W] BYTE [SHIFT] [" W] [ENTER]. The value BYTE is now stored in the variable B$. To make sure of this, type in B [SHIFT] [$ R] [ENTER]. The screen shows BYTE. This time the display is on the left side of the screen, instead of the right.

Variables handled by the **SHARP COMPUTER** are divided into the following;

- Variables
  - Numeric variables
    - Fixed numeric variables (A to Z)
    - Simple numeric variables (AB, C1, etc.)
    - Numeric array variables
  - String variables
    - Fixed character variables (A$ to Z$)
    - Simple character variables (BB$, C2$, etc.)
    - Character array variables

Fixed Variables

The first section, fixed variables, is always used by the computer for storing data. It can be thought of as pre-allocated variable space. In other words, no matter how much memory your program uses up, you will always have at least 26 variables to choose from to store data in. This data can be one of two types: NUMERIC or STRING (alphabetic character). Fixed memory locations are eight bytes long and can be used for only one type of data at a time. To illustrate this, type in the following examples:

A = 123 [ENTER]
A$ [ENTER]

You get the message:

ERROR 9

This means that you have put numeric data into the area of memory called A and then told the computer to show you that information again as STRING data. This confuses the computer so it says that there is an error condition. Press the [C-CE] key to clear the error condition. Now try the following example:

A$ = "ABC" [ENTER]
A [ENTER]

Again, the computer is confused and gives the ERROR 9 message. Look at the figure shown below to see that the variable name A equals the same area in memory as the variable name A$, and that B equals B$, and so on for all the letters of the alphabet.
Figure:

```
A = A$ = A(1)  = A$(1)
B = B$ = A(2)  = A$(2)
C = C$ = A(3)  = A$(3)
D = D$ = A(4)  = A$(4)
E = E$ = A(5)  = A$(5)
F = F$ = A(6)  = A$(6)
G = G$ = A(7)  = A$(7)
H = H$ = A(8)  = A$(8)
I = I$ = A(9)  = A$(9)
J = J$ = A(10) = A$(10)
K = K$ = A(11) = A$(11)
L = L$ = A(12) = A$(12)
M = M$ = A(13) = A$(13)
N = N$ = A(14) = A$(14)
O = O$ = A(15) = A$(15)
P = P$ = A(16) = A$(16)
Q = Q$ = A(17) = A$(17)
R = R$ = A(18) = A$(18)
S = S$ = A(19) = A$(19)
T = T$ = A(20) = A$(20)
U = U$ = A(21) = A$(21)
V = V$ = A(22) = A$(22)
W = W$ = A(23) = A$(23)
X = X$ = A(24) = A$(24)
Y = Y$ = A(25) = A$(25)
Z = Z$ = A(26) = A$(26)
```

Simple Variables

Simple variable names are specified by two or more alphanumeric characters, such as AA or B1. Unlike fixed variables, simple variables have no dedicated storage area in the memory. The area for simple variables is automatically set aside (within the program and data area) when a simple variable is first used.

Since separate memory areas are defined for simple numeric variables and simple character variables even if they have the same name, variables such as AB and AB$, for example, may be used at the same time.

Whereas alphanumeric characters are usable for simple variable names, the first character of a variable name must always be alphabetic and uppercase. If more than two characters are used to define a variable name, only the first two characters are meaningful.

Notes:
- The functions and BASIC commands inherent to the computer are not usable as variable names.
  e.g., PI, IF, TO, ON, SIN, etc.
- Each simple character variable can hold up to 16 characters and symbols.

Array Variables

For some purposes, it is useful to deal with numbers as an organized group, such as a list of scores or a tax table. In BASIC these groups are called **arrays**. An array can be either **one-dimensional**, like a list, or **two-dimensional**, like a table.

To define an array, the DIM (short for dimension) statement is used. Arrays must always be "declared" (defined) before they are used (not like the single-value variables we have been using). The form for the numeric DIMension statement is:

DIM numeric-variable-name (size)

where:

numeric-variable-name is a variable name which conforms to the normal rules for numeric variable names previously discussed.

size is the number of storage locations and must be a number in the range 0 through 255. Note that when you specify a number for the size, you get one more location than you specified.

Examples of legal numeric DIMension statements are:

```
DIM X(5)
DIM AA(24)
DIM Q5(0)
```

The first statement creates an array X with 6 storage locations. The second statement creates an array AA with 25 locations. The third statement creates an array with one location and is actually rather silly since (for numbers at least), it is the same as declaring a single-value numeric variable.

It is important to know that array variable X and variable X are separate, and unique to SHARP. The first X denotes a series of numeric storage locations, and the second a single and different location.

Now that you know how to create arrays, you might be wondering how to refer to each storage location. Since the entire group has only one name, the way in which we refer to a single location (called an "element") is to follow the group name with a number in parentheses. This number is called "subscript". Thus, for example, to store the number 8 into the fifth element of our array X (declared previously), we would write:

X(4) = 8

If the use of 4 is puzzling, remember that the numbering of elements begins at zero and continues through the size number declared in the DIM statement.

The real power of arrays lies in the ability to use an expression or a variable name as a subscript.

To declare a character array, a slightly different form of the DIM statement is used:

DIM character-variable-name (size) * length

where:

character-variable-name is a variable name which conforms to the rules for normal character variables as discussed previously.

size is the number of storage locations and must be in the range 0 to 255. Note that when you specify a number, you get one more location than you specified.

*length is optional. If used, it specifies the length of each of the strings that compose the array. Length is a number in the range 1 to 80. If this clause is not used, the strings will have the default length of 16 characters.

Examples of legal character array declarations are:

```
DIM X$(4)
DIM NM$(10)*10
DIM IN$(1)*80
DIM R$(0)*26
```

The first example creates an array of five strings, each able to store 16 characters. The second DIM statement declares an array NM with eleven strings of 10 characters each. Explicit definition of strings smaller than the default helps to conserve memory space. The third example declares a two-element array of 80-character strings and the last example declares a single string of twenty-six characters.

Besides the simple arrays we have just studied, the computer allows "two-dimensional" arrays. By analogy, a one-dimensional array is a list of data arranged in a single column. A two-dimensional array is a table of data with rows and columns. The two-dimensional array is declared by the statement:

DIM numeric-variable-name (rows, columns)

or

DIM character-variable-name (rows, columns)*length

where:

rows specifies the number of rows in the array. This must be a number in the range 0 through 255. Note that when you specify the number of rows you get one more row than the specification.

columns specifies the number of columns in the array. This must be a number in the range 0 through 255. Note that when you specify the number of columns you get one more column than the specification.

The following diagram illustrates the storage locations that result from the declaration DIM T (2, 3) and the subscripts (now composed of two numbers) that pertain to each storage location:

	Column 1	Column 2	Column 3	Column 4
Row 0	T (0, 0)	T (0, 1)	T (0, 2)	T (0, 3)
Row 1	T (1, 0)	T (1, 1)	T (1, 2)	T (1, 3)
Row 2	T (2, 0)	T (2, 1)	T (2, 2)	T (2, 3)

Note: Two-dimensional arrays can rapidly eat up storage space. For example, an array with 25 rows and 35 columns uses 875 storage locations!
Arrays are very powerful programming tools.

The following table shows the number of bytes used to define each variable and the number used by each program statement.

Variable	Variable name	Data	
Numeric variable	7 bytes	8 bytes	
String variable	7 bytes	Array variable	Specified number
		Simple variable (two-character variable)	16 bytes

* For example, if DIM Z$ (2,3)*10 is specified, 12 variables, each capable of storing 10 characters, are reserved. This requires 7 bytes (variable name) + 10 bytes (number of characters) × 12 = 127 bytes.

Element	Line number	Statement & function	Others, ENTER
Number of bytes used	3 bytes	1 byte or 2 bytes	1 byte

Variables in the Form of A( )

Whereas a data area on the computer's memory is set aside for fixed variables, it may also be used to define subscripted variables which have the same form as array variables.

There are 26 fixed variable names available, i.e., A through Z (A$ through Z$). Each of these names can be subscripted with numbers 1 through 26, such as A(1) – A(26) or A$(1) – A$(26). This means that variable A(1) may be used in place of variable A, A(2) in place of B, A(3) in place of C, and so forth.

However, if an array named A or A$ has already been defined by the DIM statement, subscripted variables named A cannot be defined. For example, if array A is defined by DIM A(5), the locations for A(0) through A(5) are set aside in the program/data area. So if you specify variable A(2), it does not refer to fixed variable B, but refers to the array variable A(2) defined in the program/data area. If you specify A(9), it will cause an error since A(9) is outside the range of the dimension specified by the DIM A(5) statement.

In turn, if subscripted variables are already defined in the form of A( ), it is not possible to define array A or A$ by using the DIM statement, unless the definition for the subscripted variables is cleared with the CLEAR statement.

* Using subscripts in excess of 26:

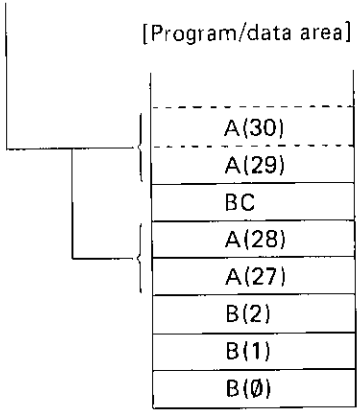
If subscripts greater than 26 are used for subscripted variables A( ) when array A is not defined by a DIM statement, the corresponding locations in the program/data area are set aside for these A( ) variables. For instance, if you execute A(35) = 5, locations for variables A(27) to A(35) will be reserved in the program/data area.

While variables subscripted in excess of 26 are treated as array variables, they are subject to the following special restrictions:

(1) Locations for an array with the same name must be contiguous in the program/data area. Otherwise, an error will occur.

```
1Ø DIM B(2)
2Ø A(28)=5
3Ø BC=12
4Ø A(30)=9
```

If this program is executed, the array named "A" is not defined in two consecutive segments in the program data area, and an error will result at line 40.



(2) Numeric array variables and character array variables with the same subscript cannot be defined at the same time. For example, A(30) and A$(30) cannot be defined at the same time as they use the same location in the program/data area.

(3) Two dimensional arrays cannot be defined, nor is it possible to specify the length of character strings to be held in character array variables. For example, the length of a character string which can be held in character array variable A$( ) is limited to seven characters or less.

(4) Variables subscripted with zero (0) cannot be defined. If A(0) or A$(0) is defined, an error will result.

(5) When subscripts greater than A(27) or A$(27) are first used, 7 bytes are used for the variable name, and 8 bytes are used for each variable.

Expressions

An **expression** is some combination of variables, constants, and operators which can be evaluated to a single value. The calculations which you entered in Chapter 3 were examples of expressions. Expressions are an intrinsic part of BASIC programs. For example, an expression might be a formula that computes an answer to some equation, a test to determine the relationship between two quantities, or a means to format a set of strings.

Numeric Operators

The computer has five **numeric operators.** These are the arithmetic operators which you used when exploring the use of the computer as a calculator in Chapter 3:

+ Addition
– Subtraction
* Multiplication
/ Division
^ Power

A **numeric expression** is constructed in the same way that you entered compound calculator operations. Numeric expressions can contain any meaningful combination of numeric constants, numeric variables, and these numeric operators:

```
(A * B) ^ 2
A(2,3) + A (3,4) + 5.0-C
(A/B) * (C+D)
```

String Expressions

String expressions are similar to numeric expressions except that there is only one string operator – concatenation (+). This is the same symbol as used for plus. When used with a pair of strings, the + attaches the second string to the end of the first string and makes one longer string. You should take care in making more complex string concatenations and other string operations because the work space used by the computer for string calculations is limited to only 79 characters.

Note: String quantities and numeric quantities cannot be combined in the same expression unless one uses one of the functions which convert a string value into a numeric value or vice versa:

```
"15" + 10 is illegal
"15" + "10" is "1510", not "25"
```

Relational Expressions

A relational expression compares two expressions and determines whether the stated relationship is True or False. The relational operators are:

> Greater Than
>= Greater Than or Equal To
= Equals
<> Not Equal To
<= Less Than or Equal To
< Less Than

The following are valid relational expressions:

```
A < B
C(1,2)> = 5
D(3)<>8
```

If A was equal to 10, B equal to 12, C(1,2) equal to 6, and D(3) equal to 9, all of these relational expressions would be True.
Chapter strings can also be compared in relational expressions. The two strings are compared character by character according to their ASCII value starting at the first character (see Appendix B for ASCII values). If one string is shorter than the other, a 0 or NUL will be used for any missing positions. All of the following relational expressions are True:

```
"ABCDEF" = "ABCDEF"
"ABCDEF" <> "ABCDE"
"ABCDEF" > "ABCDE"
```

Relational expressions evaluate to either True or False. The computer represents True by a 1; False is represented by a 0. In any logical test, an expression which evaluates to 1 or more will be regarded as True, whereas one which evaluates to 0 or less will be considered False. Good programming practice, however, dictates the use of an explicit relational expression instead of relying on this coincidence.

Logical Expressions

Logical expressions are relational expressions which use operators AND, OR, XOR, and NOT. AND, OR, and XOR are used to connect two relational expressions; the values of the combined expressions are as shown in the following tables.

A AND B

		Value of A	
		True	False
Value of B	True	True	False
	False	False	False

A OR B

		Value of A	
		True	False
Value of B	True	True	True
	False	True	False

A XOR B

		Value of A	
		True	False
Value of B	True	False	True
	False	True	False

Note: Value of A and B must be 0 (false) or 1 (true).

The XOR instruction cannot be used in combination with the AND or OR instruction in an expression. To execute expression D=(A XOR B) AND C, for example, divide the expression into two parts for execution: D=A XOR B and D=D AND C.

- Decimal numbers can be expressed in the binary notation of 16 bits as follows:

Decimal notation	16-bit binary notation
32767	0111111111111111
:	:
3	0000000000000011
2	0000000000000010
1	0000000000000001
0	0000000000000000
−1	1111111111111111
−2	1111111111111110
−3	1111111111111101
:	:
−32768	1000000000000000

The negation (NOT) of a binary number 0000000000000001 is taken as follows:

NOT	0000000000000001
(Negative)→	1111111111111110

Thus, 1 is inverted to 0, and 0 to 1 for each bit, which is called "negation (NOT operation)." Then, the following will result when 1 and NOT 1 are added together:

```
  0000000000000001 (1)
+)1111111111111110 (NOT 1) ..... ones complement
  1111111111111111 (−1)    ..... twos complement
```

Thus, all bits become 1. According to the above number list, the bits become −1 in decimal notation, that is, 1 + NOT 1 = −1.

The relationship between numerical value X and its negated (or inverted) value NOT X is:

X + NOT X = −1

This results in an equation of NOT X = −X−1, i.e.,

NOT X = −(X + 1)

From the equation, the following results are obtained:

NOT 0 = −1
NOT −1 = 0
NOT −2 = 1

More than two relational expressions can be combined with these operators. You should take care to use parentheses to make the intended comparison clear.

(A<9) AND (B>5)
(A>=10) AND NOT (A>20)
(C=5) OR (C=6) OR (C=7)
(X>=50) XOR (X<70)

The COMPUTER implements logical operators as "bitwise" logical functions on 16-bit quantities. (See note on relational expressions and True and False.) In normal operations, this is not significant because the simple 1 and 0 (True and False) which result from a relational expression uses only a single bit. If you apply a logical operator to a value other than 0 or 1, it works on each bit independently. For example, if A is 17 and B is 22, (A OR B) is 23:

17 OR 22 is 10001 .. 17
10110 .. 22 } OR operation
10111 .. 23 in decimal number

17 and 22 are first converted into binary numbers. Then for each digit, logical 1 is left if either bit is 1. Otherwise, logical 0 is left.

For example, if A is 41 and B is 27, (A XOR B) is 50:

41 XOR 27 is 101001 .. 41
011011 .. 27 } XOR operation
110010 .. 50 in decimal number

41 and 27 are first converted into binary numbers. Then, for each digit, logical 0 is left if both bits are 1 or 0.

If you are a proficient programmer, there are certain applications where this type of operation can be very useful. Beginner programmers should stick to clear, simple True or False relational expressions.

Parentheses and Operator Precedence

When evaluating complex expressions, the computer follows a predefined set of priorities which determine the sequence in which operators are evaluated. This can be quite significant:

5 + 2 * 3 could be

5 + 2 = 7
7 * 3 = 21

or

2 * 3 = 6
6 + 5 = 11

The exact rules of "operator precedence" are given in Appendix D.

To avoid having to remember all these rules and to make your program clearer, always use parentheses to determine the sequence of evaluation. The above example is clarified by writing either:

(5 + 2) * 3 or 5 + (2 * 3)

RUN Mode

In general, any of the above expressions can be used in the RUN mode well as in programming a BASIC statement. In the RUN mode an expression is computed and displayed immediately. For example:

Input	Display
(5>3) AND (2<6)	1.

The 1 means that the expression is True.

Functions

Functions are special components of the BASIC language which take one value and transform it into another value. Functions act like variables whose value is determined by the value of other variables or expressions. ABS is a function which produces the absolute value of its argument:

ABS (−5) is 5
ABS (6) is 6

LOG is a function which computes the log to the base 10 of its argument.

LOG (100) is 2
LOG (1000) is 3

A function can be used any place that a variable can be used. Many functions do not require the use of parentheses:

LOG 100 is the same as LOG (100)

You must use parentheses for functions which have more than one argument. Using parentheses always makes programs clearer.

See Chapter 9 for a complete list of functions available on the computer.

CHAPTER 5
PROGRAMMING THE COMPUTER

In the previous chapter, we examined some of the concepts and terms of the **BASIC** programming language. In this chapter, you will use these elements to create programs on the computer. Let us reiterate, however, that this is not a manual on how to program in BASIC. What this chapter will do is to familiarize you with the use of BASIC on your computer.

Programs

A **program** consists of a set of instructions to the computer. Remember the computer is only a machine. It will perform the exact operations that you specify. You, the programmer, are responsible for issuing the correct instructions.

BASIC Statements

The computer interprets instructions according to a predetermined format. This format is called a **statement**. You always enter BASIC statements in the same pattern. Statements must start with a line number:

```
1Ø: PRINT "HELLO"
2Ø: END
3Ø:
```

Line Numbers

Each line of a program must have a unique line number–any integer between 1 and 65279. Line numbers are the reference for the computer. They tell the computer the order in which to execute the program. You need not enter lines in sequential order (although if you are a beginning programmer, it is probably less confusing for you to do so). The computer always begins execution with the lowest line number and moves sequentially through the lines of a program in ascending order.

When programming, it is wise to allow increments in your line numbering (10, 20, 30, ... 10, 30, 50, etc.). This enables you to insert additional lines if necessary.
CAUTION: Do not use the same line numbers in different programs. If you use the same line number, the oldest line with that number is deleted when you enter the new line.

BASIC Verbs

All BASIC statements must contain **verbs**. Verbs tell the computer what action to perform. A verb is always contained within a program, and as such is not acted upon immediately.

```
10: PRINT "HELLO"
20: END
30:
```

Some statements require or allow an **operand**:

```
10: PRINT "HELLO"
20: END
30:
```

Operands provide information to the computer telling it what data the verb will act upon. Some verbs require operands; with other verbs they are optional. Certain verbs do not allow operands. (See Chapter 9 for a complete listing of BASIC verbs and their use on the computer.)

BASIC Commands

Commands are instructions to the computer which are entered outside of a program. Commands instruct the computer to perform some action with your program or to set modes which affect how your programs are executed.

Unlike verbs, commands have immediate effects–as soon as you complete entering the command (by pressing the [ENTER] key), the command will be executed. Commands are not preceded by a line number:

```
RUN
NEW
RADIAN
```

Some verbs may also be used as commands. (See Chapter 9 for a complete listing of BASIC commands and their use on the computer.)

Modes

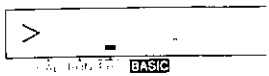
The RUN mode is used to execute the programs you create.

The PROgram mode is used to enter and edit your programs.

Beginning to Program on the Computer

After all your practice in using the computer as a calculator, you are probably quite at home with the keyboard. From now on, when we show an entry, we will not show every keystroke. Remember to use [SHIFT] to access characters above the keys and END EVERY LINE BY PRESSING THE [ENTER] KEY.

Now you are ready to program.
Slide the POWER SWITCH to the ON position and then press the [BASIC] key twice. You will see the following initial information in the display.



The above display shows that the computer is in PROgram mode.
(If a dash indicator is at the CAL or RUN label, press the [BASIC] key once or twice.)
Enter the NEW command.

Input	Display
NEW [ENTER]	>

The NEW command clears the computer's memory of all existing programs and data. The prompt appears after you press [ENTER], indicating that the computer is awaiting input.

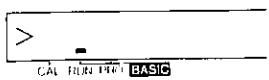
Example 1 – Entering and Running a Program

Make sure the computer is in the PRO mode and enter the following program:

Input	Display
10 PRINT "HELLO"	10 PRINT "HELLO"_

Notice that when you press [ENTER], the computer displays your input, automatically inserting a colon (:) between the line number and the verb. Verify that the statement is in the correct format.

Now press the [BASIC] key to set the RUN mode.



Input	Display
RUN [ENTER]	HELLO

Since this is the only line of the program, the computer will stop executing at this point. Press [ENTER] to get out of the program and reenter RUN if you wish to execute the program again.

Example 2 – Editing a Program

Suppose you wanted to change the message that your program was displaying, that is, you wanted to edit your program. With a single line program, you could just retype the entry, but as you develop more complex programs, editing becomes a very important component of your programming. Let's edit the program you have just written.

Are you still in the RUN mode? If so change to the PROgram mode.

You need to recall your program in order to edit it. Use the Up Arrow (↑) to recall your program. If your program was completely executed, the [↑] key will recall the last line of the program. If there was an error in the program, or if you used the BREAK ([BRK]) key to stop execution, the [↑] key will recall the line in which the error or BREAK occurred. To make changes in your program, use the [↑] key to move up in your program (recall the previous line) and the [↓] key to move down in your program (display the next line). If held down, the [↑] and [↓] keys will scroll vertically, that is, they will display each line moving up or down in your program.

You will remember that to move the cursor within a line, you use the ▶ (right arrow) and ◀ (left arrow). Using the [►] key or [◄] key, position the cursor over the first character you wish to change:

Input	Display
[↑]	10: PRINT "HELLO"
[◄][◄][◄][◄][◄]	10 PRINT "HELLO"

Notice that the cursor is now in the flashing block form indicating that it is "on top of" an existing character. Type in:

Input	Display
GOOD"!	10 PRINT"GOOD"!_

Don't forget to press [ENTER] at the end of the line. If you now change to the RUN mode by pressing [BASIC] and enter the RUN command, the following appears:

Input	Display
RUN [ENTER]	ERROR 1 IN 10

This is a new kind of error message. Not only is the error type identified (our old friend the syntax error) but the line number in which the error occurs is also indicated.

Press the [C-CE] and then return into the PROgram mode. You must be in the PROgram mode to make changes in a program. Using the [↑], recall the last line of your program.

Input	Display
[↑]	10: PRINT "GOOD" !

The flashing cursor is positioned over the problem area. In Chapter 4, you learned that when entering string constants in BASIC, all characters must be contained within quotation marks. Use the [DEL] key to eliminate the "!":

Input	Display
[SHIFT] [DEL]	10 PRINT "GOOD"_

Now let's put the "!" in the correct location. When editing programs, DELete and INSert are used in exactly the same way as they are in editing calculations (see Chapter 3). Using the [◄], position the cursor on top of the character which will be the first character following the insertion.

Input	Display
[◄]	10 PRINT "GOOD"

Press the [INS] key. A ⎵ will indicate the spot where the new data will be entered:

Input	Display
[SHIFT] [INS]	10 PRINT "GOOD⎵"

Type in the !. The display looks like this:

Input	Display
!	10 PRINT "GOOD!"

Remember to press [ENTER], so the correction will be entered into the program.

NOTE: If you wish to DELete an entire line from your program, just type in the line number and the original line will be eliminated.

Example 3 – Using Variables in Programming

If you are unfamiliar with the use of numeric and string variables in BASIC, reread these sections in Chapter 4.

Using variables in programming allows much more sophisticated use of the computer's computing abilities.

Remember, you assign numeric fixed variables using any letters from A to Z:

```
A = 5
```

To assign a string variable, you also use a letter, followed by a dollar sign. **Do not use the same letter in designating a numeric and a string fixed variable.** You cannot designate A and A$ in the same program.

Remember that each string fixed variable must not exceed 7 characters in length:

```
A$ = "TOTAL"
```

The value assigned to a variable can change during the execution of a program, taking on the value typed in or computed during the program. One way to assign a variable is to use the INPUT verb. In the following program, the value of A$ will change in response to the data typed in answer to the inquiry "WORD?".
Enter this program:

```
10 INPUT "WORD?"; A$
20 B= LEN (A$)
30 PRINT "WORD␣IS␣"; B; "␣LTRS"
40 END
```

␣ means space

Before you RUN the program, note several new features. Line 30 of this program exceeds the 24-character maximum of the computer's display. When a line is longer than 24 characters (up to the 79-character maximum), the computer moves the characters to the left as the 24-character maximum is exceeded. This does not destroy the previous input. This move to the left is referred to as horizontal scrolling.

The second new element in this program is the use of the END statement to signal the completion of a program. END tells the computer that the program is completed. It is always good programming practice to use an END statement.

As your programs become more complex, you may wish to review them before beginning execution. To look at your program, use the LIST command. LIST, which can only be used in the PROgram mode, displays programs beginning with the lowest line number.

Try listing this program:

Input	Display
LIST [ENTER]	10: INPUT "WORD?" ; A$

Use the [↑] and [↓] keys to move through your program until you have reviewed the entire program. To review a line which contains more than 24 characters, move the cursor to the extreme right of the display and the additional characters will appear on the screen. After checking your program, run it:

Input	Display
RUN [ENTER]	WORD?_
HELP [ENTER]	WORD IS 4. LTRS
[ENTER]	>

This is the end of your program. Of course, you may begin it again by entering RUN. However, this program would be a bit more entertaining if it presented more than one opportunity for input. We will now modify the program, so it will keep running without entering RUN after each answer.

Return to the PRO mode and use the up or down arrow (or LIST) to reach line 40.

You may type 40 to delete the entire line or use the [►] key to position the cursor over the E in END. Change line 40 so that it reads:

40: GOTO 10

Now RUN the modified program.

The GOTO statement causes the program to loop (keep repeating the same operation). Since you put no limit on the loop, it will keep going forever (an "infinite" loop). To stop this program, hit the BREAK ([BRK]) key.

When you have stopped a program using the [BRK] key, you can restart it using the CONT command. CONT stands for CONTinue. With the CONT command, the program will restart on the line that was being executed when the [BRK] key was pressed.

Example 4 – More Complex Programming

Although the computer has a factorial function, we will use an example of the factorial computation in this section to explain more complex programming. The following program computes N Factorial (N!). The program begins with 1 and computes N! up to the limit which you enter. Enter this program.

```
100 F = 1: WAIT 118
110 INPUT "LIMIT?"; L
120 FOR N = 1 TO L
130 F = F * N
140 PRINT N, F
150 NEXT N
160 END
```

Several new features are contained in this program. The WAIT verb in line 100 controls the length of time that displays are held before the program continues. The numbers and their factorials are displayed as they are computed. The time they appear on the display is set by the WAIT statement to approximately 2 seconds, instead of waiting for you to press [ENTER].

Also in line 100, notice that there are two statements on the same line separated by a colon (:). **You may put as many statements as you wish on one line, separating each by a colon, up to the 80-character maximum including** [ENTER]. Multiple statement lines can make a program hard to read and modify, however, so it is a good programming practice to use them only where the statements are very simple or there is some special reason to want the statements on one line.

Also in this program, we have used the FOR verb in line 120 and the NEXT verb in line 150 to create a loop. In Example 3, you created an "infinite" loop which kept repeating the statements inside the loop until you pressed the [BRK] key. With the FOR/NEXT loop, the computer adds 1 to N each time execution reaches the NEXT verb. It then tests to see if N is larger than the limit L. If N is less than or equal to L, execution returns to the top of the loop and the statements are executed again. If N is greater than L, execution continues with line 160 and the program stops.

You may use any numeric variable in a FOR/NEXT loop. You also do not have to start counting at 1 and you can add any amount at each step. See Chapter 9 for details.

We have labeled this program with line numbers starting with 100. Labeling programs with different line numbers allows you to have several programs in memory at one time. To RUN this program instead of the one at line 10 enter:

```
RUN 100
```

In addition to executing different programs by giving their starting line number, you can give programs an alphabetical name and start them with the [DEF] key (see Chapter 6).

You will notice that while the program is running, the BUSY indicator is lit at those times that there is nothing on the display. RUN the program a few more times and try setting N at several different values.

Storing Programs in Memory

Programs remain in memory when you turn off the computer, or it undergoes an AUTO OFF. Even if you use the [BRK] , [C·CE] , or [CA] key, the programs will remain in memory.

Programs are lost from memory only when you:

* enter NEW before beginning programming.
* initialize the computer using the RESET button.
* create a new program using the SAME LINE NUMBERS as a program already in memory.
* change the batteries.

This brief introduction to programming on the computer should serve to illustrate the exciting programming possibilities of your new computer.

CHAPTER 6
SHORT CUTS

The computer includes several features which make programming more convenient by reducing the number of keystrokes required to enter repetitive material.

One such feature is in the availability of abbreviations for verbs and commands (see Chapter 9).

This chapter discusses the additional feature which can eliminate unnecessary typing– the DEF key. (DEF is short for "DEFINE".)

The DEF Key and Labeled Programs

Often you will want to store several different programs in the computer memory at one time. (Remember that each must have unique line numbers.) Normally, to start a program with a RUN or GOTO command, you need to remember the beginning line number of each program (see Chapter 9). But, there is an easier way! You can label each program with a letter and execute the program using only two keystrokes. This is how to label a program and execute it using DEF:

Note: Put a label on the first line of each program that you want to reference. The label consists of a single character in quotes, followed by a colon (:).

```
10: "A": PRINT "FIRST"
20: END
80: "B": PRINT "SECOND"
90: END
```

Any one of the following characters can be used: A, S, D, F, G, H, J, K, L, ', Z, X, C, V, B, N, M, and SPC. Notice that these are the keys in the bottom two rows of the alphabetic portion of the keyboard.

Note: To execute the program, instead of typing RUN 80 or GOTO 10, you need only press the DEF key and then the letter used as a label. In the above example, pressing DEF and then 'B' would cause 'SECOND' to appear on the display.

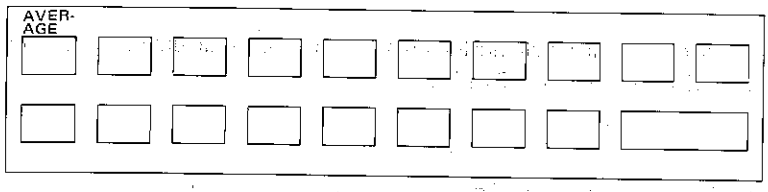
When DEF is used to execute a program, variables and mode settings are affected in the same way as when GOTO is used. See Chapter 9 for details.

Template

One template is provided with the computer. You can use this template to help you remember frequently used DEF key assignments.

For example, if you have one group of programs which you often use at the same time, label the programs with letters and mark the template and set it over the two bottom rows of the keyboard so that you can easily begin execution of any of the programs with two keystrokes.

Example:



CHAPTER 7
USING PRINTERS AND CASSETTE INTERFACE

Your computer can use the following printers, cassette interfaces, and cassette recorders as options:

- CE-126P Printer/Cassette Interface
- CE-140P Color Dot Printer
- CE-515P Color Plotter/Printer
- CE-152 Cassette Tape Recorder

This chapter describes the important points in using these options, as well as the functions of the serial I/O interface built into the computer to allow use of the optional color printers and other peripheral units.

Using CE-126P Printer/Cassette Interface

The optional **CE-126P** Printer/Cassette Interface allows you to add a printer and to connect a cassette recorder to your **SHARP COMPUTER.**
The **CE-126P** features:

* 24-column thermal printer.
* Convenient paper feed and tear bar.
* Simultaneous printing of calculation results as desired (except in the CAL mode)
* Easy control of display or printer outputs in BASIC.
* Built-in cassette interface with remote function.
* Manual and programmed control of recorder for storing programs and data
* Dry battery operation for portability.

For connecting the computer to the CE-126P, refer to the instruction manual supplied with the CE-126P.

1. Using Printer

If you are using the computer for manual calculations in BASIC mode, you may use the CE-126P to simultaneously print the results of your calculations.

CAUTION:

The results obtained by the direct calculation feature in manual calculations cannot be printed. Calculation results in the CAL mode also cannot be printed.

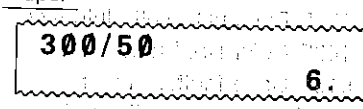
Printing is easily accomplished by pressing the [SHIFT] key and then the [ENTER] key (P↔NP) while in the RUN mode.

The printer indicator (a dash symbol) will appear just above the "PRINT" label in the lower right area of the display. After this, when you press the [ENTER] at the end of a calculation, the contents of the display will be printed on one line and the results will be printed on the next. For example:

Input

300/50 [ENTER]

Paper



You may print output on the printer from within BASIC programs by using the LPRINT statement (see Chapter 9 for details). LPRINT can be used in the same form as the PRINT statement. The difference is that if you PRINT something longer than 24 characters to the display, there is no way for you to see the extra characters. With the LPRINT verb, the extra characters will be printed on a second, and possibly a third, line as is required.

Programs which have been written with PRINT can be converted to work with the printer by including a PRINT=LPRINT statement in the program (see Chapter 9 for details). All PRINT statements following this statement will act as if they were LPRINT statements. PRINT=PRINT will reset this condition to its normal state. This structure may also be included in a program in an IF statement allowing a choice of output at the time the program is used.

You may also list your programs on the printer with the LLIST command (see Chapter 9 for details). If used without line numbers, LLIST will list all program lines currently in memory in their numerical order by line number. A line number range may also be given with LLIST to limit the lines which will be printed. When program lines are longer than 24 characters, two or more lines may be used to print one program line. The second and succeeding lines will be indented four or six characters so that the line number will clearly identify each separate program line. (Line number, 1 to 999: four-character indentation, over 999: six-character indentation)

Caution:

- In case an error (ERROR code 8) occurs due to a paper jam, remove the jam by pulling the paper toward the paper cutter and tearing off the paper. Then press the [C·CE] key to clear the error condition.

- When the printer is exposed to strong external electrical noise, it may print numbers at random. If this happens, depress the [BRK] key to stop the printing. Turn the **CE-126P** power off and on, and then press the [C·CE] key.
  Pressing the [C·CE] key will return the printer to its normal condition.
- When the printer causes a paper jam or is exposed to strong external electrical noise while printing, it may not operate normally and only the symbol "BUSY" is displayed. If this happens, depress the [BRK] key to stop printing. (Remove the paper jam.) Turn the **CE-126P** power off and on, and then press the [C·CE] key.
- When the **CE-126P** is not in use, turn off the printer switch to conserve the battery life.

2. Using Cassette Interface

Using this cassette interface will allow you to store programs and data from the computer onto cassette tape (of course, you'll also need for this pocket computer system the optional cassette tape recorder, model CE-152). Once on tape, you can load these programs and data back into the computer with a simple procedure.

Connecting the CE-126P to a tape recorder

Only these three connections are necessary:

1. Red plug into the MICrophone jack on the cassette recorder.
2. Gray plug into the EARphone jack on the cassette recorder.
3. Black plug into the REMote jack on the cassette recorder.

3. Cassette Tape Recorder

We recommend that you use an optional cassette tape recorder **CE-152** for your computer system. The **CE-152** designed to match the **COMPUTER** records programs and data via the **CE-126P** cassette interface. Any recorded program can be retrieved and reloaded into the **COMPUTER**.

If you intend to use a cassette tape recorder other than the **CE-152**, the following are the minimum tape recorder requirements for interfacing with the **CE-126P**:

Item	Requirements
1. Recorder Type	Any tape recorder, standard cassette or microcassette recorder, may be used in accordance with the requirements outlined below.
2. Input Jack	The recorder should have a minijack input labeled "MIC". Never use the "AUX" jack.
3. Input Impedance	The input jack should be a low impedance input (200~1,000 ohms).
4. Minimum Input level	Below 3 mV or −50 dB
5. Output Jack	Should be a minijack labeled "EXT.(EXTernal speaker)", "MONITOR" "EAR (EARphone)" or equivalent.
6. Output Impedance	Should be below 10 ohms.
7. Output Level	Should be above 1 V (practical maximum output above 100 mV).
8. Distortion	Should be within 15% within a range of 2 kHz through 4 kHz.
9. Wow and Flutter	0.3% maximum (WRMS)
10. Others	Recorder motor speed should not fluctuate.

* In case the miniplug provided with the **CE-126P** is not compatible with the input/output jacks of your tape recorder, special line conversion plugs commercially available may be used.

Note: Some tape recorders may reject connection due to different specifications. Those tape recorders having distortion, increased noise, and power deterioration after long years of use may not show satisfactory results owing to change in their electrical characteristics.

4. Operating the Cassette Interface and Recorder

Recording (saving) onto magnetic tape

See Tape Notes.

(1) Turn off the REMOTE switch on the CE-126P.

(2) Enter a program or data into the computer.

(3) Load a tape into the tape recorder.
Determine the position on the tape where you want to record the program.
- When using a tape, be sure the tape moves past the clear leader (nonmagnetic mylar material).
- When using a tape already partially recorded, search for a location where no recording exists.

(4) Connect the Interface's red plug to the tape recorder's MIC jack and the black plug to the REM jack.

(5) Turn on the REMOTE switch.

(6) Simultaneously press the RECORD and PLAY buttons on the tape recorder (to put it in RECORD mode).

(7) Input recording instructions (CSAVE statement, PRINT# statement), and press the [ENTER] key for execution.

First set the computer in the "RUN" or "PRO" mode. Next operate the following keys: [C] [S] [A] [V] [E] [SHIFT] ["] file name [SHIFT] ["] [ENTER].
(To write the contents of data memory onto tape, operate keys as follows: e.g., [P] [R] [I] [N] [T] [SHIFT] [#] [ENTER].
e.g., [C] [S] [A] [V] [E] [SHIFT] ["] [A] [A] [SHIFT] ["] [ENTER])

When you press the [ENTER] key, tape motion will begin, leaving about an 8-second no-signal blank. (A long pip sounds for a while at the beginning.) After that, the file name and its contents are recorded (with continuous short beep sounds).

(8) When the recording is complete, the PROMPT symbol (>) will be displayed and the tape recorder will automatically stop. Now you have your program on tape (it still is in the computer also).
When data is to be automatically recorded by program execution (PRINT# statement, not manual operation), set up steps (1) thru (6) before executing the program.

To aid you in locating programs on tapes, use the tape counter on the recorder.

Verifying the computer and tape contents

See Tape Notes.

After loading or transferring a program to or from tape, you can verify that the program on tape and program in the computer are identical (and thus be sure that everything is OK before continuing your programming or execution of programs).

(1) Turn off the REMOTE switch.

(2) With cassette in the recorder, operate the tape motion controls to position tape at the point just before the appropriate file name to be checked.

(3) Connect the gray plug to the EARphone and the black plug to the REMote jacks.

(4) Turn on the REMOTE switch.

(5) Press the PLAY button of the recorder.

(6) Input a CLOAD? statement and start execution with [ENTER] key. Do this as follows: Set the computer in the "RUN" or "PRO" mode.

Enter the following keys:—

The file name which you used previously.

[C] [L] [O] [A] [D] [SHIFT] [?] [SHIFT] ["] [A] [A] [SHIFT] ["] [ENTER]

The computer will automatically search for the specified file name and will compare the contents on tape with the contents in memory.

During the verification, the mark "*" is shown at the rightmost digit of the display. The "*" will disappear when the verification is completed. While a file name is being retrieved, no "*" will be displayed as the verification has not started yet.
(The same occurs when the first program is read without a file name.)

If the programs are verified as being identical, a PROMPT symbol (>) will be displayed on the computer.

If the programs differ, execution will be interrupted and Error code 8 will be displayed. If this occurs, try again.

Loading from a magnetic tape

See Tape Notes.

To load, transfer, or read out programs and data from magnetic tape into the computer, use the following procedure.

(1) Turn off the REMOTE switch.

(2) Load the tape in the tape recorder. Move the tape to a point just before the portion to be read out.

(3) Connect the gray plug to the EAR jack on the tape recorder, and the black plug to the REM jack.
In using a tape recorder having no REM terminal, press the PAUSE button to make a temporary stop.

(4) Turn on the REMOTE switch.

(5) Push the PLAY button on the tape recorder (to put unit in playback mode).

Set the VOLUME control to middle or maximum.
If the tape recorder does not function properly when the volume is set to maximum, turn the volume down and try again.
Set Tone to maximum treble.

(6) Input transfer instructions (CLOAD statement, INPUT# statement), and press [ENTER] key for execution.

Put the computer in the "RUN" mode. Then operate the following keys:
[C] [L] [O] [A] [D] [SHIFT] ["] file name [SHIFT] ["] [ENTER] .
(To load the contents of the data memory, operate keys as follows:
e.g., [I] [N] [P] [U] [T] [SHIFT] [#] [ENTER] .
e.g., [C] [L] [O] [A] [D] [SHIFT] ["] [A] [A] [SHIFT] ["] [ENTER] .)

The specified file name will be automatically searched for and its contents will be transferred into the computer.
The "*" appears while loading the designated CSAVEd program from the tape to the computer's memory.
(The same occurs when the first program is read without a file name.)
The "*" disappears when the load is performed completely.

(7) When the program has been transferred, the computer will automatically stop the tape motion and display the PROMPT (>) symbol.

To transfer data (INPUT# statement) in the course of a program, set up steps (1) thru (5) prior to executing the program.

Notes:
- If an error occurs (error code "8" is displayed), start over from the beginning.
  If the error continues, adjust volume slightly up or down.
- If the error code is not displayed but the tape motion continues (while the computer is displaying the symbol "BUSY"), transfer is not being properly executed.
  Press [ON BRK] key ("break") to stop the tape. Repeat steps from the beginning.
- If the error remains or the tape continues to run after several attempts to correct the problem, try cleaning and demagnetizing the recorder's tape head.

Tape Notes

(1) For any transfer or verification, use the tape recorder that was used for recording. If another tape recorder is used, transfer or verification may not be possible.

(2) Always use only the highest quality tape for programs and data storage (economy grade audio type tape may not provide the proper characteristics for digital recordings).

(3) Keep the tape heads and other parts that contact tape clean. Use a cassette cleaner tape for this purpose.

(4) Volume setting – Set to middle or maximum level
Volume level can be very important when reading in data from the recorder; make slight adjustments as required to obtain error-free data transfer. A slight adjustment either up or down may help produce perfect results every time.

(5) Be sure all connections between the computer and cassette interface are secure. And be sure the connections between interface and recorder are secure and free of foreign matter.

(6) If problems occur when using AC power for the **CE-126P** and/or the recorder, use battery power instead (sometimes the AC power connection adds some "hum" to the signal which may upset proper digital recording).

- To connect the AC adaptor to the **CE-126P**, turn the **CE-126P** power off and then connect the adaptor to the **CE-126P**.

(7) Tone control – Set to maximum treble.

(8) When recording programs or data on used tape, erase some beginning portion of the used type before writing and then execute the BASIC command for recording. (Make sure that the previous program is completely erased without any portion remaining.)

Using Color Printers

Connection of the optional CE-140P Color Dot Printer or CE-515P Color Plotter/Printer to the **COMPUTER** allows you to have hard-copy outputs of programs and calculation results as well as graphic printouts in multiple colors. The CE-140P can draw a figure in seven different colors using four color cartridge inks, whereas the CE-515P draws a chart or diagram with four different color pens. The CE-140P printer can be connected directly to the serial I/O interface of the computer. However, the CE-515P requires cable (CE-516L) for connection to the serial I/O interface.

Color Printer Notes

The BASIC commands used in the **COMPUTER** include various printer commands applicable to a color dot printer or color plotter/printer, as well as the LLIST and LPRINT commands for printing programs and calculation results.

The CE-515P color plotter/printer can be used only when the serial input/output interface of the **COMPUTER** is in the "Open" state. Therefore, be sure to execute the OPEN command before executing any printer command to the plotter/printer.

Drawing Range and Coordinates of Graphics

When printing data in graphic form with the CE-140P or CE-515P, you may use a printer function called scissoring, whereby that portion of a figure or diagram that should fall on the printing paper is actually drawn and that portion that should fall outside the printing paper is imaginarily drawn. This function is very convenient when you wish to draw only a part of a figure, but it is easier for you to prepare a program for drawing the entire figure, or when you draw a large diagram by dividing it into some parts according to the size of the printing paper.

However, if your program is incorrect, the figure intended to be drawn on the printing paper may be drawn in the imaginary area. So be sure to program correctly when you use this function.

The directions and positions required for drawing a line in graphics printing are expressed by X and Y coordinates. With regard to coordinate X for horizontal direction, "−" indicates the leftward direction and "+", the rightward direction. With regard to coordinate Y for vertical direction, "+" indicates the upward direction and "−", the downward direction.

When using any of the optional color printers, refer to the operation manual supplied with each printer for detailed information.

Serial I/O Function

The **COMPUTER** is equipped with a serial I/O interface. This interface function can be used to connect the optional CE-140P or CE-515P color printer to the **COMPUTER** for graphic printing in multiple colors and to allow data transfer between the **COMPUTER** and the host computer.

Note: Exercise care since applying to the I/O terminal a voltage exceeding the allowable range of the **COMPUTER** may damage the internal parts.

Basics on Use of the Serial I/O Interface

The circuit of the serial I/O interface is usually closed. If closed, data from the serial I/O terminal cannot be sent and the received data cannot be read.
Therefore, you must open the circuit using the OPEN command. (An attempt to execute this command when already open will result in ERROR 8.)

Further, the conditions to perform data I/O with the host computer connected to the **COMPUTER** must be satisfied. In other words, the conditions for I/O signals must be the same for both the **COMPUTER** and the connected host. If the conditions are different, the I/O signals (data) cannot be read correctly and this will result in data errors. The OPEN command can be used to set or modify the I/O conditions.
After the conditions for both sending and receiving sides are satisfied and the circuit opened, the following commands are used to perform data or program I/O.

LPRINT, LLIST, SAVE, LOAD, PRINT #1, INPUT #1

At the end of the data (or program) transfer, the circuit of the serial I/O interface is closed. Although the CLOSE command is used to close the circuit, the circuit is also closed when the program ends (such as when the END command is executed) or when the RUN command is executed.
When writing a program which uses the serial I/O interface, the circuit must be opened, the I/O operation performed, and then the circuit must be closed as described above.

Note: The **COMPUTER** is not equipped with a timer function to interrupt communication with the connected equipment by measuring the waiting time for each I/O command to the serial I/O interface.
Therefore, if the connected equipment is not ready to communicate (such as when the power is off) while a command is being executed or if communication at the connected equipment is interrupted, the **COMPUTER** cannot terminate its command execution.
If this is the case, press the BRK key and stop the command execution.

Serial I/O Function during CE-140P Use

Connection of the COMPUTER with the optional CE-140P color dot matrix printer enables on-the-spot printout with a single compact unit.

A serial I/O interface is provided on the side of the CE-140P printer to enable connection of the computer and printer to another personal computer.

* The CE-140P switch should normally be set to SIO. The printer can be used as follows depending on the system configuration.

CE-140P switch position	Serial I/O interface	CE-140P only connected	CE-140P and CE-126P connected*1
SIO (Set here for normal use.)	OPEN	I/O performed from CE-140P I/O interface	Same as left
	CLOSED	Output printed on CE-140P	Output printed on CE-126P*3
PRINTER	OPEN*2	Output printed on CE-140P	Same as left
	CLOSED	Output printed on CE-140P	Output printed on CE-126P*3

*1 In this case, the CE-126P printer is connected to the connector on the left side of the **COMPUTER** together with the CE-140P printer.

*2 When using the CE-140P in this setting, specify serial I/O conditions with the OPEN command as described below (Initialized: refer to the OPEN command.)
OPEN "1200, N, 8, 1, A, C, &1A"
If the I/O condition setting is not correct, the CE-140P printer will not operate properly.

*3 Commands related to graphics are output to the CE-140P.

Notes: 1. Before connecting the CE-140P to the computer, be sure to turn off the power of the computer.
2. While the LOW BATTERY indicator of the CE-140P is illuminating, an attempt to produce any outputs on the CE-140P by commands from the computer will result in an error (Error code 8).
3. When the CE-140P is in the ink replaceable state, the power of the computer cannot be turned off even if you slide the power switch to the OFF position. Press the [INK] key of the CE-140P and then turn off the power of the computer.

CHAPTER 8
USING THE RAM CARD

The **COMPUTER** stores all programs and data (fixed variables, simple variables, string variables, others) in the RAM (Random Access Memory) card. RAM card CE-212M (8K bytes) is an option, but one card has already been included in your computer as a built-in accessory. Six other optional RAM cards are also available and can be easily installed in the **COMPUTER**. These RAM cards offer storage capacities ranging from 2K bytes to 32K bytes.
Use of a larger-capacity RAM card expands computer program and data storage area quickly and easily. Furthermore, multiple RAM cards can be interchangeably used to store programs not currently needed.

The program written into the RAM card is retained (by the battery inserted into the RAM card) even if the RAM card is removed from the computer. Therefore, the same program can be used by reinserting the RAM card into the computer.

RAM CARD Options

Card size	Type	Capacity	Program/data area	Data retention period
1/2	CE-210M	2K bytes	734 bytes	40 months
	CE-211M	4K bytes	2,782 bytes	24 months
	CE-212M	8K bytes	6,878 bytes	15 months
	CE-2H16M	16K bytes	15,070 bytes	14 months
Regular	CE-201M	8K bytes	6,878 bytes	34 months
	CE-202M	16K bytes	15,070 bytes	18 months
	CE-203M	32K bytes	31,454 bytes	18 months

Notes: 1. 1/2-size card types are about half the size of the regular-size card types.
2. RAM card can be also used as an exclusive data area or system area.
3. "Data retention period" refers to an approximate duration for which the contents of the RAM card with a new battery can be retained when the RAM card is removed from the computer and stored.

RAM Card Replacement

- Do not remove the RAM card from the computer except when replacing the card with another RAM card or when replacing the RAM card battery or main batteries.
- After an unused RAM card has been installed or when the RAM card removed for battery replacement has been reinstalled, be sure to perform the following: Turn off the power of the computer and press the RESET button and the following message will appear on the display.

BUSY
MEMORY ALL CLEAR O.K.?

Then press the [ENTER], [Y], or [=] key. This will clear all the previous contents, if any, of the RAM card.

The stored program and data are cleared when the battery in the RAM card is replaced (or removed). Therefore, if a valuable program is stored in the RAM card, it is recommended that the program be recorded on tape beforehand.

As for the regular size RAM card, however, if the battery in the RAM card is replaced while the RAM card is installed in the **COMPUTER**, the contents are not cleared.

Note: Do not press the RESET button on the rear of the **COMPUTER** while the RAM card is installed.

The program and data stored within the RAM card are cleared if the RESET button is pressed. If it becomes necessary to press the RESET button, refer to page 10, and press it while holding down the appropriate key.

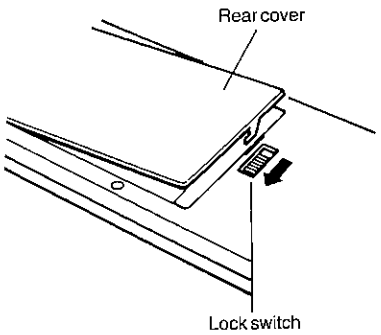
1. Replacing 1/2-Size RAM Card

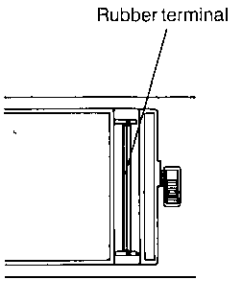
If the replacement RAM card is to be used for the first time, be sure to insert the battery into the RAM card.

(1) Turn the power of the **COMPUTER** off.

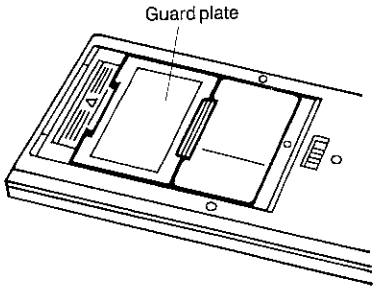
(2) Slide the rear cover lock switch in the arrow direction shown in the figure and remove the rear cover.

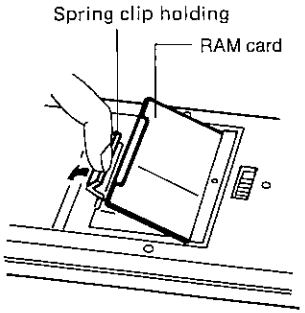
Note: Do not touch the circuits, components, rubber terminals, etc., found within the computer.





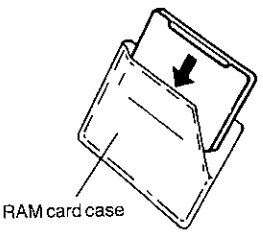
(3) Remove the guard plate and then remove the RAM card while lifting the spring clip holding the RAM card in place.



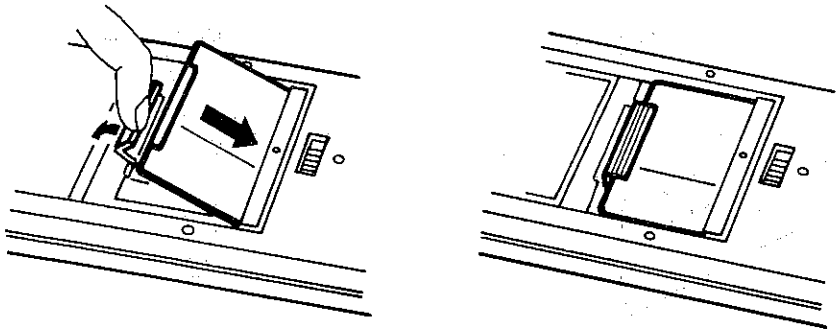


(4) Take out another RAM card from its case, and keep the replaced RAM card in the RAM card case.

Note: When replacing the 1/2-size card with a regular-size card, keep the replaced 1/2-size card in the regular-size card case.

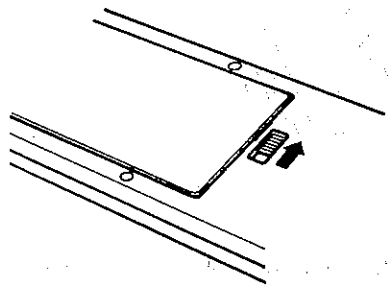


(5) Lift the spring clip and insert the RAM card into the RAM slot with the terminals down. Insert at an angle so the terminals fit into the slot, then lower the back until the card lies flat. Release the spring clip.



Note: Do not insert the card upside down. Improper RAM card insertion could cause the card and computer to short, resulting in serious damage to both.

(6) Cover the RAM card with the guard plate, attach the rear cover to the computer, and slide the lock switch to the "LOCK" position.



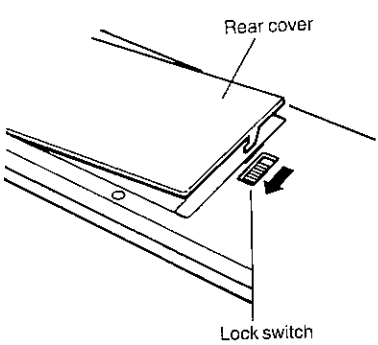
Note: Be sure to slide the lock switch to the "LOCK" position. Otherwise, the computer will not operate. If the power is turned on with the lock switch unlocked, be sure to lock the switch and turn the power off. Then turn the power on again to operate the computer.

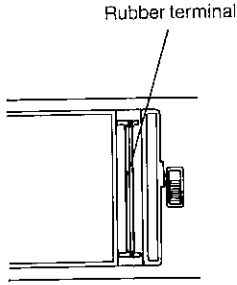
2. Replacing Regular-Size RAM Card

If the replacement RAM card is to be used for the first time, be sure to insert the battery into the RAM card.

(1) Turn off the power of the computer.

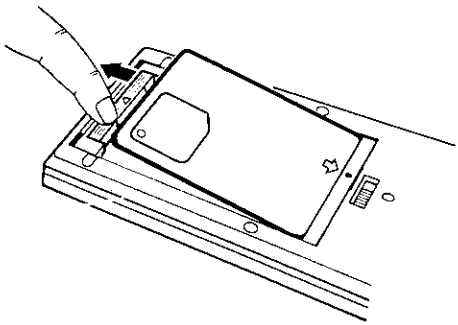
(2) Slide the rear cover lock switch in the arrow direction shown in the figure and remove the rear cover.





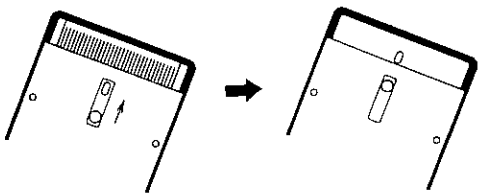
Note: Do not touch the circuits, components, rubber terminals, etc., found within the computer.

(3) Pull the card mounting/removal lever in the arrow direction shown in the figure. This releases the RAM card from the tab so that it can be removed.

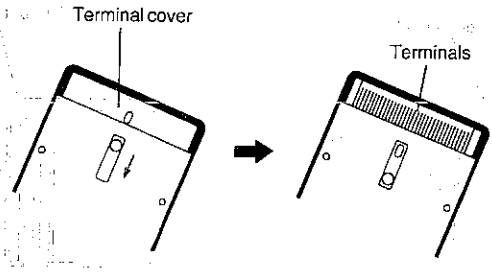


(4) Close the terminal cover of the replaced RAM card upon its removal.

Note: Be sure to close the terminal cover completely.

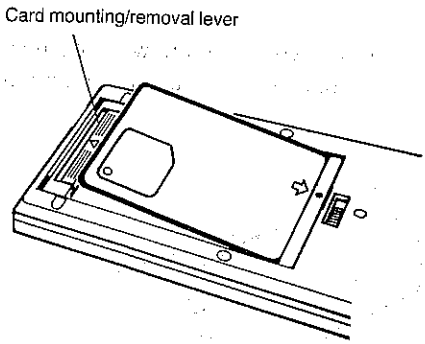


(5) Take out another RAM card from its case and open the terminal cover of the RAM card.



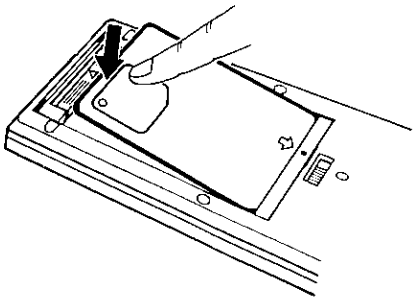
Note: Completely open the terminal cover (to bring the terminals into full view), but do not touch the terminals.

(6) Insert the terminal end of the RAM card down into the computer as shown in the figure.



Note: Do not insert the RAM card backwards nor insert it with the terminal cover not completely open. Doing so may short the terminals and damage the computer.

(7) Then, gently press the battery cover. The card will snap onto the tab of the card mounting/removal lever.



(8) Attach the rear cover and slide the lock switch to the "LOCK" position.

Note: Be sure to slide the lock switch to the "LOCK" position. Otherwise, the computer will not operate. If the power is turned on with the lock switch unlocked, be sure to lock the switch and turn the power off. Then turn the power on again to operate the computer.

Using RAM Card with Other Models

The RAM cards used with the **COMPUTER** can also be used with PC-1350, PC-1450, PC-2500, or PC-1100. RAM cards used with these models can also be used with the **COMPUTER**.

(1) When not using programs interchangeably
After the RAM card has been installed, turn the power on and then push the RESET button on the back of the computer without pressing any key.

(2) When using programs interchangeably

1) Programs written on other models
These programs can be used as they are. However, be sure to note the following:

* Some modifications of the program are required if it uses any commands not available with the **COMPUTER**.

* Previously input data in the computer cannot be used for the program execution as they will all be erased.

* The following message will appear in the display when the program is too large to be handled with the **COMPUTER**.

BUSY
MEMORY ALL CLEAR O.K.?

Should this message appear, just turn the power off. The program will be retained. To erase the program, press [ENTER], [Y], or [=] key while the message is on display.
(The **COMPUTER** must be able to secure data area and system area within the RAM card. When this cannot be done, the above message will appear in the display.)
If no key is pressed for about 2 minutes after the above "CLEAR O.K.?" message, the COMPUTER will be automatically put in the AUTO OFF state.

2) Programs written on the **COMPUTER**

With the RAM card mounted on the **COMPUTER**, operate CONVERT [ENTER] in the RUN mode. The program(s) in the RAM card will be automatically converted into programs executable by other models and the display of the **COMPUTER** will disappear, indicating the end of the conversion. (During the conversion, do not press the [BRK] key. Otherwise, you must again perform the conversion process from the beginning.) Note that RAM card CM-203M can be used with the COMPUTER only. An error will result if you attempt to convert the programs in the CE-203M.

* Programs containing commands not available to PC-1350, PC-1450, PC-2500, or PC-1100 must be modified.

* Previously input data in the other model cannot be used and will all be erased.

CHAPTER 9
BASIC REFERENCE

This chapter is divided into three sections:

Commands: Instructions which are used outside a program to change the working environment, perform utilities, or control programs

Verbs: Action words used in programs to construct BASIC statements

Functions: Special operators used in BASIC programs to change one variable to another

However, serial I/O related commands, graphics-related verbs, and text functions are summarized in the following pages.

Graphics related verbs ..... pages 195 ~ 213
Serial I/O related commands .... pages 225 ~ 240
Text functions ..... pages 241 ~ 243

Commands and verbs are arranged alphabetically within each category in the respective sections. Each entry is on a separate page for easy reference. Functions are grouped into four categories and arranged alphabetically within each category. The contents of each section are listed on the following three pages so that you can quickly identify the category to which an operator belongs.

Commands

Program Control
- CONT
- DELETE
- GOTO*
- NEW
- RENUM
- RUN

Cassette Control
- CLOAD
- CLOAD?
- CSAVE
- INPUT#*
- MERGE
- PRINT#*

Debugging
- LIST
- LLIST
- TROFF*
- TRON*

Serial I/O
- CLOSE#1
- CONSOLE
- INPUT#1
- LLIST
- LOAD
- LPRINT
- OPEN
- OPEN$
- PRINT#1
- SAVE

Variables Control
- CLEAR*
- DIM*
- MEM*

Angle Mode Control
- DEGREE*
- GRAD*
- RADIAN*

Graphic Printing
- CIRCLE*
- COLOR*
- CROTATE*
- CSIZE*
- GLCURSOR*
- GRAPH*
- LF*
- LLINE*
- LTEXT*
- PAINT*
- RLINE*
- SORGN*

Others
- BEEP*
- MDF*
- PASS
- RANDOM*
- USING*
- WAIT*

* These commands are also BASIC verbs. Their effect as commands is identical to their effect as verbs so they are not described in the Commands section. See the Verbs section for more information.

Verbs

Control and Branching
- CHAIN
- END
- FOR...TO...STEP
- GOSUB
- GOTO
- IF...THEN
- NEXT
- ON...GOSUB
- ON...GOTO
- RETURN
- STOP

Assignment and Declaration
- CLEAR
- DIM
- LET

Input and Output
- AREAD
- CSAVE
- DATA
- INPUT
- INPUT#
- INPUT#1
- LOAD
- LPRINT
- PAUSE
- PRINT
- PRINT#
- PRINT#1
- READ
- RESTORE
- USING
- WAIT

Graphic Printing
- CIRCLE
- COLOR
- CROTATE
- CSIZE
- GLCURSOR
- GRAPH
- LF
- LLINE
- LTEXT
- PAINT
- RLINE
- SORGN

Others
- BEEP
- CLOSE #1
- CONSOLE
- DEGREE
- GRAD
- MDF
- OPEN
- RADIAN
- RANDOM
- REM
- TROFF
- TRON

Functions

Pseudovariables

- INKEY$
- MEM
- PI

String Functions

- ASC
- CHR$
- LEFT$
- LEN
- MID$
- RIGHT$
- STR$
- VAL

TEXT Functions

- BASIC
- TEXT

Numeric Functions

- ABS
- ACS
- AHC
- AHS
- AHT
- ASN
- ATN
- COS
- CUR
- DEG
- DMS
- EXP
- FACT
- HCS
- HSN
- HTN
- INT
- LN
- LOG
- POL
- RCP
- REC
- RND
- ROT
- SGN
- SIN
- SQR
- SQU
- TAN
- TEN

COMMANDS

1 **CLOAD**
2 **CLOAD** "filename"

Abbreviations: CLO., CLOA.

See also: CLOAD?, CSAVE, MERGE, PASS

Purpose

The CLOAD command is used to load a program saved on cassette tape.

Use

The first format of the CLOAD command clears existing programs in memory and loads the first program stored on the tape, starting at the current position.

The second format of the CLOAD command clears the memory, searches the tape for the program whose name is given by "filename", and loads the program.

If the computer is in PROgram or RUN mode, program memory is loaded from the tape.

Examples

CLOAD	Loads the first program from the tape.
CLOAD "PRO3"	Searches the tape for the program named 'PRO3' and loads it.

Notes: 1. If the designated file name cannot be retrieved before the tape reaches the end, the computer will continue to search the file name. In this case, stop the retrieval function by pressing the ON BRK key. This applies to MERGE, CHAIN, CLOAD? and INPUT# commands to be described later.
2. If an error occurs during execution of CLOAD or CHAIN command (described later), the program stored in the computer will be invalid.

- During the loading, an asterisk "*" is shown at the far right digit position of display. The "*" will disappear when the loading is completed. While a file name is being retrieved, no "*" will be displayed as the loading has not started yet. (The same occurs when the first program is read without a file name.)

1 **CLOAD?**
2 **CLOAD?** "filename"

Abbreviations: CLO.?, CLOA.?

See also: CLOAD, CSAVE, MERGE, PASS

Purpose

The CLOAD? command is used to compare a program saved on cassette tape with one stored in memory.

Use

To verify that a program was saved correctly, rewind the cassette tape to the beginning of the program and use the CLOAD? command.
The first format of the CLOAD? command compares the program stored in memory with the first program stored on the tape, starting at the current position.

The second format of the CLOAD? command searches the tape for the program whose name is given by "filename" and then compares it to the program stored in memory.

Examples

CLOAD?	Compares the first program from the tape with the one in memory.
CLOAD? "PRO3"	Searches the tape for a program named 'PRO3' and compares it to the one stored in memory.

* An asterisk "*" will appear at the far right digit position of the display while the program is being verified. The asterisk will disappear and the prompt will reappear when verification is completed.

1 **CONT**

Abbreviations: C., CO., CON.

See also: RUN, STOP verb

Purpose

The CONT command is used to continue a program which has been temporarily halted.

Use

When the STOP verb is used to halt a program during execution, the program can be continued by entering CONT in response to the prompt.

When a program is halted using the BRK key, the program can be continued by entering CONT in response to the prompt.

CONT also functions when the program is temporarily interrupted due to a command such as PRINT.

Examples

CONT Continues an interrupted program execution.

1 **CSAVE**
2 **CSAVE** "filename"
3 **CSAVE**, "password"
4 **CSAVE** "filename", "password"

Abbreviations: CS., CSA., CSAV.

See also: CLOAD, CLOAD?, MERGE, PASS.

Purpose

The CSAVE command is used to save a program to cassette tape.

Use

The first format of the CSAVE command writes all of the programs in memory onto the cassette tape without a specified file name.

The second format of the CSAVE command writes all of the programs in memory onto the cassette tape and assigns the indicated file name.

The third format of the CSAVE command writes all of the programs in memory onto the cassette tape without a specified file name and assigns the indicated password. Programs saved with a password may be loaded by anyone, but only someone who knows the password can list or modify the programs. (See discussion under PASS command.)

The fourth format of the CSAVE command writes all of the programs in memory onto the cassette tape and assigns them the indicated file name and password.

Examples

CSAVE "PRO3", "SECRET" — Saves the programs now in memory onto the tape under the name 'PRO3', protected with the password 'SECRET'.

Note: When saving a program to cassette tape via the CE-124 cassette interface with the **COMPUTER** connected to the CE-140P, use the **COMPUTER** after turning off the power of the CE-140P. Saving a program from the computer without turning off the CE-140P will put the computer in the AUTO OFF state immediately after the saving operation.
If you wish to perform the verification of the saved data, press the ON/BRK key to turn on the power of the **COMPUTER** and then use the computer in BASIC (RUN or PRO) mode for verification.

1 **DELETE** [starting line number][,[ending line number]]

Abbreviations: DEL., DELE., DELET.

See also: NEW, PASS

Purpose

The DELETE command is used to delete a program line or program lines. This command is effective for manual operation in the PROgram mode.

Use

If both the starting and ending line numbers are specified, all the program lines between the two numbers inclusive are deleted. If either of the two specified line numbers is not found, an error occurs.

If only the starting line number is specified, that line number alone is deleted.

If the starting line number and comma (,) are specified, the starting line number and all the subsequent line numbers are deleted.

If the comma (,) and ending line number are specified, all the lines from the beginning of the program to the ending line number are deleted.

If both the starting and ending line numbers are omitted, an error occurs.

When the DELETE command is executed while a program file is being read based on the MERGE command, the DELETE command works on the program last read.

The DELETE command is ignored when any password is set in the program.

Example

DELETE 1ØØ, Deletes line 100 and all the subsequent line numbers.

1 **GOTO** expression

Abbreviations: G., GO., GOT.

See also: RUN

Purpose

The GOTO command is used to start the execution of a program.

Use

The GOTO command can be used in place of the RUN command to start program execution at the line number specified by the expression.

GOTO differs from RUN in six respects:

1) The value of the interval for WAIT is not reset.
2) The display format established by USING statements is not cleared.
3) Variables and arrays are preserved.
4) PRINT=LPRINT status is not reset.
5) The pointer for READ is not reset.
6) The serial I/O circuit is not closed.

Execution of a program with GOTO is identical to execution with the DEF key.

Examples

GOTO 1ØØ Begins program execution at line 100.

1 **LIST**
2 **LIST** line number
3 **LIST** "label"

Abbreviations: L., LI., LIS.

See also: LLIST

Purpose

The LIST command is used to display a program.

Use

The LIST command may only be used in the PROgram mode.

* With format 1, the program is displayed from its first line until the display is full.
* With format 2, the program is displayed from the line of the specified line number until the display is full.
  If the line for the specified number does not exist, the program will be displayed from the line with the next largest number which does exist.
* With format 3, the program is displayed from the line written with the specified label until the display is full.
* When programs are merged with the MERGE command, the LIST command functions for the last program.
  However, if the label specified in format 3 does not exist in the last program, it is searched for in sequence from the first program. If the specified label is found, the line containing it is displayed.
* If a password has been set, the LIST command is ignored.

Examples

```
LIST 100
```
Displays line number 100.

1 **LLIST**
2 **LLIST** expression
3 **LLIST** expression 1, expression 2
4 **LLIST** expression,
5 **LLIST**, expression

Abbreviations: LL., LLI., LLIS.

See also: LIST

Purpose

The LLIST command is used for printing a program on the optional printer.

Use

* When the serial I/O interface is open according to the OPEN command, the LLIST command outputs the program to the serial I/O interface terminal of the printer. (See page 228.)
  To return the program print command to the printer, execute the CLOSE command.

The LLIST command may be used in the PROgram or RUN mode.

The first format prints all of the programs in memory.

The second format prints only the program line whose line number is given by the expression.

The third format prints the statements from the line number with the nearest line equal to or greater than the value of expression 1 to the nearest line equal to or greater than the value of expression 2. There must be at least two lines between the two numbers.

The fourth format prints all program lines beginning with the line whose number is given by the expression.

The fifth format prints all program lines up to, and including, the line whose number is given by the expression.

* When programs are merged with the MERGE command, the LLIST command functions for the last program. To list a program stored earlier, execute:

```
LLIST "label",
```

If a password has been set, the LLIST command is ignored.

- For printing a program on the CE-140P printer, set the number of columns to be printed per line to 24 or more. (Refer to the CONSOLE command.)
  An error occurs if the number of columns to be printed per line is set to less than 24 and the LLIST command is executed (ERROR 3).
- For printing the program on the CE-140P printer, "$\sqrt{\ }$" will be converted to SQR and "π" to PI.

Example

LLIST 100,200 Lists the statements between line numbers 100 and 200.

Note: LLINE, RLINE, GLCURSOR, LTEXT, GRAPH, LF, CSIZE, COLOR, SORGN, CROTATE, CIRCLE, or PAINT verb should be written at the beginning of a program line following the line number. If any of these verbs is written at any position after the 24th character of the program line, the desired graphic data will not be properly output on the CE-140P printer.

1 **MERGE**
2 **MERGE** "filename"
(effective for the manual operation in the PROgram or RUN mode)

Abbreviations: MER., MERG.

See also: CLOAD

Purpose

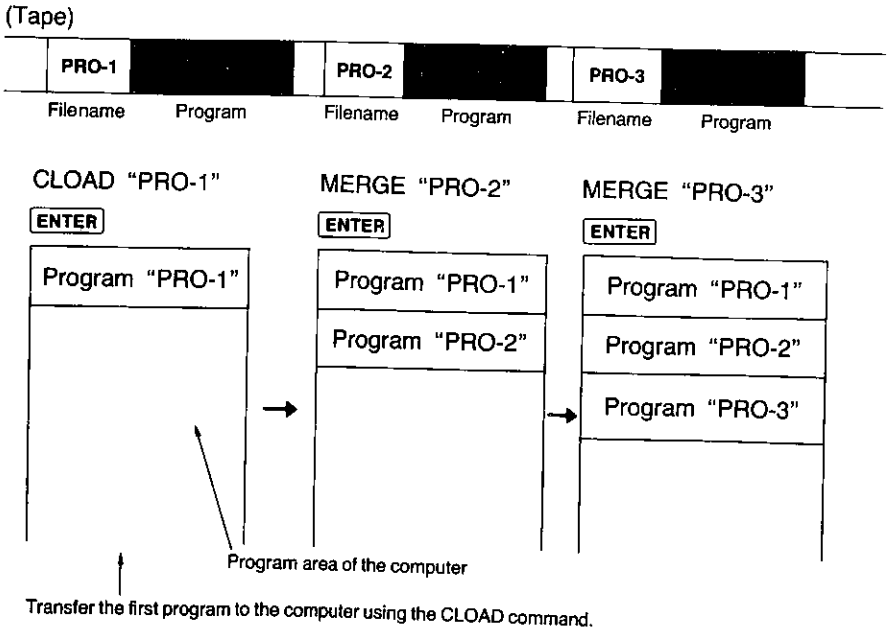
The MERGE command is used to load a program saved on cassette tape and merge it with the program existing in memory.

Use

The MERGE command retains the program already stored in the **COMPUTER** and then loads a program recorded on the tape. Therefore, several different programs can be stored in the **COMPUTER** at the same time.

Example

When programs with file names PRO-1, PRO-2, and PRO-3 are to be stored, PRO-1 is stored using the CLOAD command, whereas PRO-2 and PRO-3 are transferred to the computer using the MERGE command. The state of the storage is as follows.



Programs loaded using the MERGE command are stored as in the example. The programs are handled by their line numbers as follows.

- If the first line number of the program loaded using the MERGE command is larger than the last line number of the previously loaded program, the two programs are considered to be a single program.
- If the first line number of the program loaded using the MERGE command is smaller than the last line number of the previously loaded program, the two programs are considered separate.
  In the example above, where the line numbers for programs PRO-1, PRO-2, and PRO-3 are 10 to 200, 50 to 150, and 160 to 300, respectively, PRO-1 and PRO-2 are considered separate. PRO-2 and PRO-3 are considered to be a single program with line numbers 50 to 300.

* Loading programs with the MERGE command may result in two or more programs in the computer with the same line numbers. In this case, the executed RUN or GOTO (RUN expression, GOTO expression) is valid only for the last merged program. There will be no way to execute the preceding program(s).
Therefore, put a label to the beginning of a program to be executed and execute the program using a DEF (defined) key.
Note, however, that only the last merged program can be edited after the MERGE command has been executed and that the program(s) loaded earlier cannot be edited. Therefore, add the label to the program before merging the next program.

Merging password-protected programs

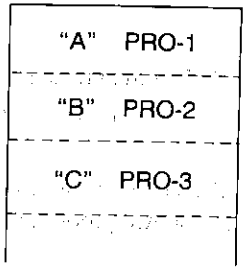
When loading programs with passwords (password-protected programs) using the MERGE command, the handling of the programs differs as outlined below, depending on whether the programs within the computer are protected.

When protected
Password-protected programs cannot be loaded.

When not protected
If password-protected programs are loaded using the MERGE command, all programs within the computer become protected.

When the programs within the computer are protected, even programs without passwords become password-protected when loaded using the MERGE command.

Executing merged programs



The figure shows the memory when PRO-1 is loaded after which PRO-2 and PRO-3 are loaded using the MERGE command. If a program is started using RUN or GOTO (RUN expression or GOTO expression), PRO-3 will be executed. On the other hand, if started using RUN "label", GOTO "label", or a DEF (defined) key, the specified label is searched for from the beginning of PRO-3 within the computer.

If not found in PRO-3, the search proceeds in PRO-1. If also not found in PRO-1, PRO-2 is searched. If the label is found, the program is executed from the labeled line.

Note that since the label is searched for in this manner, if a label used in PRO-1 and PRO-2 is also used in PRO-3, PRO-1 and PRO-2 cannot be executed.

1 **NEW**

Abbreviations: none

See also: CLEAR, PASS

Purpose

The NEW command is used to clear existing programs and data in memory.

Use

When used in the PROgram mode, the NEW command clears all programs and data (array variables, simple variables, and fixed variables) which are currently in memory. (The programs with passwords cannot be cleared.)

The NEW command is not defined in the RUN mode and will result in an ERROR 9.

Examples

NEW	Clears programs and data in memory.

1 **PASS** "character string"

Abbreviations: PA., PAS.

See also: CLOAD, CSAVE, DELETE, NEW, RENUM

Purpose

The PASS command is used to set and cancel passwords.

Use

Passwords are used to protect programs from inspection or modification by other users. A password consists of a character string no more than seven characters long. The seven characters may be alphabetic or one of the following special symbols:

! # $ % & ( ) * + - / : ; < = > ? @ $\sqrt{\ }$ π ^

Note: Do not use any BASIC command or verb as a password.

Once a PASS command has been given, the programs in memory are protected. A password-protected program cannot be examined or modified in memory. It cannot be sent to tape or listed with LIST or LLIST, nor is it possible to add or delete program lines. If several programs are in memory and PASS is entered, all programs in memory are protected. The only way to remove this protection is to execute another PASS command with the same password.

Note: When a password with more than seven characters is declared, only the first seven characters are valid and are set or removed from protection.
Press [ENTER] right after the password.
Writing characters or symbols after a password results in an error and the password cannot be canceled.
(example) PASS"ABCDEFG":A=123 [ENTER] → Error 1

Examples

PASS "SECRET" Establishes the password 'SECRET' for all programs in memory.

When a password is set and the RAM card is removed, all the programs stored in the RAM card are protected.

1 **RENUM** [new line number][,[old line number][,increment]]

Abbreviations: REN., RENU.

Purpose

The RENUM command is used to renumber program lines. This command is effective for manual operation in the PRO (Program) mode.

Use

This command renumbers old line numbers in the specified step increments, starting from the specified new line number.

If the values of new line number and increment are omitted, 10 is assumed for both. If old line number is omitted, renumbering starts from the first line of the program. If the specified old line number is not found, an error occurs.
Example 1: RENUM
Renumbers all the program lines in increments of 10 steps from line 10.
Example 2: RENUM 1Ø0, 5Ø, 1Ø
Changes old line number 50 to new line number 100 and renumbers subsequent line numbers in increments of 10 steps.

The RENUM command automatically changes all line number references following GOTO, GOSUB, IF~THEN, ON~GOTO, ON~GOSUB, RESTORE, etc., to reflect the new line numbers. In this case, however, an error will result if expression (e.g., GOTO 2*50). If an error occurs due to such incorrect line number reference, renumbering of the incorrect line number cannot be effected by RENUM. In such a case, temporarily rewrite the command containing an incorrect line number to a REM statement, and correct it (perhaps, change to ON~GOTO) after the execution of the RENUM command.

The RENUM command cannot be executed if the number of lines to be renumbered exceeds 65279, or if the specification requires a change in the execution order of program lines (for example, an attempt is made to execute RENUM 15, 30 when three program lines 10, 20, and 30 exist).

It will take a little while to complete the execution of RENUM on a large program. If you press the BRK key to interrupt the program while one asterisk (*) is appearing at the rightmost end of the display, the program will return to the original condition before the execution of RENUM. However, this interruption by the BRK key will be ignored when two asterisks (**) are on the display.

The work area of "number of program lines × 4 bytes" is used only when the RENUM command is executed. By renumbering program lines, line number references by GOTO, GOSUB, etc., also change. As a result, the original program may have an increase in the number of bytes used. In other words, new line GOTO 200 uses one byte more than old line GOTO 20. The RENUM command cannot be executed if the remaining capacity of the work area becomes short due to the increase in the number of bytes used. In such a case, clear variables from memory by the CLEAR command and you may be able to execute RENUM.

(See APPENDIX A for error messages related to RENUM.)

1 **RUN**
2 **RUN** line number

Abbreviations: R., RU.

See also: GOTO, MERGE

Purpose

The RUN command is used to execute a program in memory.

Use

The first format of the RUN command executes a program beginning with the lowest numbered statement in memory.

The second format of the RUN command executes a program beginning with the specified line number.

* When programs are merged with the MERGE command, the last merged program will be executed with format 1 or "RUN expression" in format 2.

RUN differs from GOTO in six respects:

1) The value of the interval for WAIT is reset.
2) The display format established by USING statements is cleared.
3) Variables and arrays other than the fixed variables are cleared.
4) PRINT=PRINT status is set.
5) The pointer for READ is reset to the beginning DATA statement.
6) Closes the serial I/O circuit (serial port).

Execution of a program with GOTO is identical to execution with the DEF key. In all three forms of program execution, FOR/NEXT and GOSUB nesting is cleared.

Examples

RUN 1ØØ Executes the program which begins at line number 100.

Verbs

1 **AREAD** variable name

Abbreviations: A., AR., ARE., AREA.

See also: INPUT verb and discussion of the use of the DEF key in Chapter 6

Purpose

The AREAD verb is used to read in a single value to a program which is started using the DEF key.

Use

When a program is labeled with a letter so that it can be started using the DEF key, the AREAD verb can be used to enter a single starting value without the use of the INPUT verb. The AREAD verb must appear on the first line of the program following the label. If it appears elsewhere in the program, it will be ignored. Either a numeric or string variable may be used, but only one can be used per program.

To use the AREAD verb, type the desired value in the RUN mode, press the DEF key, followed by the letter which identifies the program. If a string variable is being used, it is not necessary to enclose the entered string in quotes.

Examples

```
10 "X": AREAD N
20 PRINT N^2
30 END
```

Entering "7 DEF X" will produce a display of "49".

Notes: 1. When the display indicates PROMPT (">") at the start of program execution, the designated variable is cleared.

2. When the contents are displayed by PRINT verb at the start of program execution, the following is stored:

Example: When the program below is executed;
10 "A": PRINT "ABC", "DEFG"
20 "S": AREAD A$: PRINT A$
RUN mode
DEF [A] → ABC DEFG
DEF [S] → DEFG

- When the display indicates PRINT Numeric expression, Numeric expression, or PRINT "String", "String", the contents displayed last are stored.
- When the display indicates PRINT Numeric expression; Numeric expression; Numeric expression..., the contents displayed first (on the extreme left) are stored.
- When the display indicates PRINT "String"; "String"; "String"..., the contents of the "String" designated last (on the extreme right) are stored.

1 **BEEP** expression

Abbreviations: B., BE., BEE.

Purpose

The BEEP verb is used to produce an audible tone.

Use

The BEEP verb causes the **COMPUTER** to emit one or more audible tones at 4 kHz. The number of beeps is determined by the expression, which must be numeric (positive number less than 9.999999999E+99). The expression is evaluated, but only the integer part is used to determine the number of beeps.

BEEP may also be used as a command using numeric literals and predefined variables. In this case, beeps occur immediately after the ENTER key is pressed.

Examples

```
1Ø A=5 B$="9"
2Ø BEEP 3          Produces 3 beeps.
3Ø BEEP A          Produces 5 beeps.
4Ø BEEP(A+4)/2     Produces 4 beeps.
5Ø BEEP B$         This is illegal and will produce an ERROR 9 message.
6Ø BEEP -4         Produces no beeps, but does not produce an error message.
```

1 **CHAIN**
2 **CHAIN** expression
3 **CHAIN** "filename"
4 **CHAIN** "filename", expression

Abbreviations: CHA., CHAI.

See also: CLOAD, CSAVE, and RUN

Purpose

The CHAIN verb is used to start the execution of a program which has been stored on cassette tape. It can only be used in connection with the optional CE-126P and CE-152.

Use

To use the CHAIN verb, one or more programs must be stored on a cassette. Then, when the CHAIN verb is encountered in a running program, a program is loaded from the cassette and executed.

The first format of CHAIN loads the first program stored on the tape and begins execution with the lowest line number in the program. The effect is the same as having entered CLOAD and RUN when in the RUN mode.

The second format of CHAIN loads the first program stored on the tape and begins execution with the line number specified by the expression.

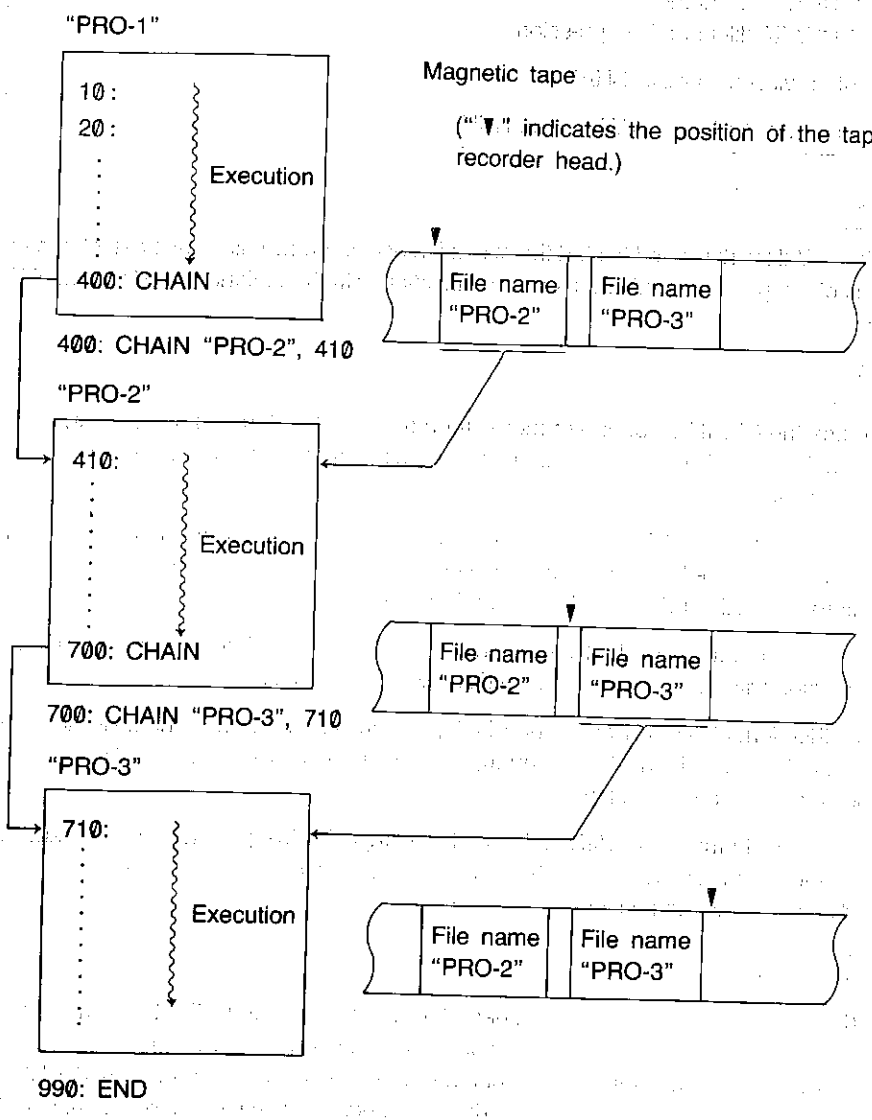
The third format of CHAIN searches the tape for the program whose name is indicated by the filename, loads the program, and begins execution with the line number indicated by the expression.

The fourth format of CHAIN will search the tape for the program whose name is indicated by the filename, load the program, and begin execution with the line number indicated by the expression.

Examples

1Ø CHAIN	Loads the first program from the tape and begins execution with the lowest line number.
2Ø CHAIN "PRO-2", 48Ø	Searches the tape for a program named PRO-2, loads it, and begins execution with line number 480.

For example, let's assume you have three program sections named PRO-1, PRO-2, PRO-3. Each of these sections ends with a CHAIN statement.



During execution, when the computer encounters the CHAIN statement, the next section is called into memory and executed. In this manner, all of the sections are eventually run.

Note: When a program containing a CHAIN command is loaded from a tape by the MERGE command, check to be sure that the CHAIN command is correct.

1 **CLEAR**

Abbreviations: CL., CLE., CLEA.

See also: DIM

Purpose

The CLEAR verb is used to erase all variables which have been used in the program and to reset all preallocated variables to zero or null.

Use

The CLEAR verb recovers space which is being used to store variables. This might be done when the variables used in the first part of a program are not required in the second part and availables space is limited. CLEAR may also be used at the beginning of a program when several programs are resident in memory and you want to clear out the space used by execution of prior programs.

CLEAR does not free the space used by the variable A–Z, A$–Z$, or A(1)–A(26) (without DIM declaration) since they are permanently assigned (see Chapter 4). CLEAR does reset numeric variables to zero and string variables to null.

Examples

```
1Ø A=5: DIM C(5)
2Ø CLEAR
```

Frees the space assigned to C( ) and resets A to zero.

1 **DATA** expression list
Where: expression list is: expression
or: expression, expression list

Abbreviations: DA., DAT.

See also: READ, RESTORE

Purpose

The DATA verb is used to provide values for use by the READ verb.

Use

When assigning initial values to an array, it is convenient to list the values in a DATA statement and use a READ statement in a FOR...NEXT loop to load the values into the array. When the first READ is executed, the first value in the first DATA statement is returned. Succeeding READs use succeeding values in the sequential order in which they appear in the program, regardless of how many values are listed in each DATA statement or how many DATA statements are used.

DATA statements have no effect if encountered in the course of regular execution of the program, so they can be inserted wherever it seems appropriate. Many programmers like to include them immediately following the READ which uses them. If desired, the values in a DATA statement can be read a second time by using the RESTORE statement.

Examples

```
10 DIM B(10)
20 WAIT 128
30 FOR I=1 TO 10
40 READ B(I)
50 PRINT B(I)
60 NEXT I
70 DATA 10,20,30,40,50,60
80 DATA 70,80,90,100
90 END
```

Sets up an array.

Loads the values from the DATA statement into B( ). B(1) will be 10, B(2) will be 20, B(3) will be 30, etc.

1 **DEGREE**

Abbreviations: DE., DEG., DEGR., DEGRE.

See also: GRAD and RADIAN

Purpose

The DEGREE verb is used to specify the unit of angle to decimal degrees.

Use

The **COMPUTER** has three forms for representing values in angular units–decimal degrees, radians, and grads. These forms are used in specifying the arguments to the SIN, COS, and TAN functions and in returning the results from the ASN, ACS, and ATN functions.

The DEGREE function changes the unit of angle for all values to decimal degrees until a GRAD or RADIAN verb is used. The DMS and DEG functions can be used to convert values in decimal degrees into sexagesimal equivalent (degrees, minutes, seconds) and vice versa.

Examples

```
1Ø DEGREE
2Ø X=ASN 1      X now has a value of 90, i.e., 90 degrees, the arc sine of 1.
3Ø PRINT X
```

1 **DIM** dim list

Where: dim list	is: dimension spec.
	or: dimension spec., dim list
and: dimension spec.	is: numeric dim spec.
	or: string dim spec.
and: numeric dim spec	is: numeric name (size)
and: string dim spec	is: string name (dims)
	or: string name (dims)*len
and: numeric name	is: valid numeric variable name
and: string name	is: valid string variable name
and: dims	is: size
	or: size, size
and: size	is: number of elements
and: len	is: length of each string in a string array

Abbreviations: D., DI.

Purpose

The DIM is used to reserve space for numeric and string array variables.

Use

Except for arrays in the form: A( ), A$( ), and simple variable like A1 or B2$, a DIM verb must be used to reserve space for any array variable.

The maximum number of dimensions in any array is two; the maximum size of any one dimension is 255. In addition to the number of elements specified in the dimension statement, one additional "zeroth" element is reserved. For example, DIM B(3) reserves B(0), B(1), B(2), and B(3). In two dimensional arrays there is an extra "zeroth" row and column.

In string arrays, one specifies the size of each string element in addition to the number of elements. For example, DIM B$(3)*12 reserves space for 4 strings which are each a maximum of 12 characters long. If the length is not specified, each string can contain a maximum of 16 characters.

When a numeric array is dimensioned, all values are initially set to zero; in a string array, the values are set to null.

For the array A and A$ DIM declaration, refer to the paragraph discussing variables.

Array variables can be cleared (or set undefined) with the CLEAR command. When the program is started using the RUN command, array variables are automatically cleared.

The variable name once declared cannot be declared again. When a program once executed is executed again with the GOTO command on using the DEF key, the same variable name as formerly declared will be declared again if the line with the DIM command is executed. In this case, clear the array variable with the CLEAR command and then declare it again.

Examples

```
10 DIM B(10)
20 DIM C$(4, 4)*10
```

Reserves space for a numeric array with 11 elements.
Reserves space for a two dimensional string array with 5 rows and 5 columns; each string will be a maximum of 10 characters.

1 **END**

Abbreviations: E., EN.

Purpose

The END verb is used to signal the end of a program.

Use

When multiple programs are loaded into memory at the same time, a mark must be included to indicate where each program ends so that execution does not continue from one program to another. This is done by including an END verb as the last statement in the program.

When the serial I/O circuit is opened by the OPEN command, the circuit is closed. (Refer to the CLOSE command.)

Examples

```
1Ø PRINT "HELLO"
2Ø END
3Ø PRINT "GOODBYE"
4Ø END
```

With these programs in memory a 'RUN 10' prints 'HELLO', but not 'GOODBYE'. 'RUN 30' prints 'GOODBYE'.

1 **FOR** numeric variable=expression 1 **TO** expression 2
2 **FOR** numeric variable=expression 1 **TO** expression 2
STEP expression 3

Abbreviations: F. and FO.; STE.

See also: NEXT

Purpose

The FOR verb is used in combination with the NEXT verb to repeat a series of operations a specified number of times.

Use

The FOR and NEXT verbs are used in pairs to enclose a group of statements which are to be repeated. The first time this group of statements is executed, the loop variable (the variable named immediately following the FOR) has the value of expression 1.

When execution reaches the NEXT verb, the loop variable is increased by the step size and then this value is tested against expression 2. If the value of the loop variable is less than or equal to expression 2, the enclosed group of statements is executed again, starting with the statement following the FOR. In the first form, the step size is 1; in the second form, the step size is given by expression 3. If the value of the loop variable is greater than expression 2, execution continues with the statement which immediately follows the NEXT. Because the comparison is made at the end, the statements within a FOR/NEXT pair are always executed at least once.

Expression 1, expression 2, and expression 3 must be in the range of −9.999999999E99 to 9.999999999E99. If the value of expression 3 is zero, FOR/NEXT loop will be infinite.

The loop variable may be used within the group of statements, for example, as an index to an array, but care should be taken in changing the value of the loop variable.

Programs should be written so that they never jump from outside a FOR/NEXT pair to a statement within a FOR/NEXT pair. Similarly, programs must never leave a FOR/NEXT pair by jumping out. Always exit a FOR/NEXT loop via the NEXT statement. To do this, set the loop variable to a value higher than expression 2.

The group of statements enclosed by a FOR/NEXT pair can include another pair of FOR/NEXT statements which use a different loop variable as long as the enclosed pair is completely enclosed; i.e., if a FOR statement is included in the group, the matching NEXT must also be included. FOR/NEXT pairs may be "nested" up to five levels deep.

Examples

```
1Ø FOR I=1 TO 5
2Ø PRINT I
3Ø NEXT I
```

This group of statements prints the numbers 1, 2, 3, 4, 5.

```
4Ø FOR N=1Ø TO Ø STEP -1
5Ø PRINT N
6Ø NEXT N
```

This group of statements counts down 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0.

```
7Ø FOR N=1 TO 1Ø
8Ø X=1
9Ø FOR F=1 TO N
1ØØ X=X*F
11Ø NEXT F
12Ø PRINT X
13Ø NEXT N
```

This group of statements computes and prints N factorial for the numbers from 1 to 10.

Note: The execution of the FOR-NEXT loop does to the end even if it jumps out of the loop. Therefore, note that a nesting error of the FOR-NEXT loop (ERROR 5) may result depending on the program (programs which execute the FOR command a number of times).

1 **GOSUB** expression

Abbreviations: GOS., GOSU.

See also: GOTO, ON..GOSUB, ON...GOTO, RETURN

Purpose

The GOSUB verb is used to execute a BASIC subroutine.

Use

When you wish to execute the same group of statements several times in the course of a program or use a previously written set of statements in several programs, it is convenient to use the BASIC capability for subroutines using the GOSUB and RETURN verbs.

The group of statements is included in the program at some location where they are not reached in the normal sequence of execution. A frequent location is after the END statement which marks the end of the main program. At those locations in the main body of the program–where subroutines are to be executed–include a GOSUB statement with an expression which indicates the starting line number of the subroutine. The last line of the subroutine must be a RETURN. When GOSUB is executed, the **COMPUTER** transfers control to the indicated line number and processes the statements until a RETURN is reached. Control is then transferred back to the statement following the GOSUB.

A subroutine may include a GOSUB. Subroutines may be "nested" in this fashion up to 10 levels deep.

The expression in a GOSUB statement may not include a comma, e.g., 'A(1, 2)' cannot be used. Since there is an ON...GOSUB structure for choosing different subroutines at given locations in the program, the expression usually consists of just the desired line number. When a numeric expression is used, it must evaluate to a valid line number, i.e., 1 to 65279, or an ERROR 4 will occur.

Example

```
10 GOSUB 100
20 END
100 PRINT "HELLO"
110 RETURN
```

When this program is run it prints the word 'HELLO' one time.

1 **GOTO** expression

Abbreviations: G., GO., GOT.

See also: GOSUB, ON...GOSUB, ON...GOTO

Purpose

The GOTO verb is used to transfer control to a specified line number.

Use

The GOTO verb transfers control from one location in a BASIC program to another location. Unlike the GOSUB verb, GOTO does not "remember" the location from which the transfer occurred.

The expression in a GOTO statement may not include a comma, e.g., 'A(1,2)' cannot be used. Since there is an ON...GOTO structure for choosing different destinations at given locations in the program, the expression usually consists of just the desired line number, i.e., 1 to 65279, or an ERROR 4 will occur.

Well designed programs usually flow simply from beginning to end, except for subroutines executed during the program. Therefore, the principal use of the GOTO verb is as a part of an IF...THEN statement.

Examples

```
10 INPUT A$.
20 IF A$="Y" THEN GOTO 50
30 PRINT "NO"
40 GOTO 60
50 PRINT "YES"
60 END.
```

This program prints 'YES' if a 'Y' is entered and prints 'NO' if anything else is entered.

1 **GRAD**

Abbreviations: GR., GRA.

See also: DEGREE and RADIAN

Purpose

The GRAD verb is used to specify the unit of angle to grads.

Use

The **COMPUTER** has three forms for representing values in angular units–decimal degrees, radians, and grads. These forms are used in specifying the arguments to the SIN, COS, and TAN functions and in returning the results from the ASN, ACS, and ATN functions.

The GRAD function changes the unit of angle for all values to grads until a DEGREE or RADIAN verb is used. Grad represents an angular measurement in terms of percent gradient, i.e., a 45° angle is a 50^{g} gradient.

Examples

To solve for the values of a sine, in the respective angular units, first specify "D" for degrees, "R" for radians, or "G" for grads and then enter the angle of the sine.

```
10 INPUT "DEG=D, RAD=R, GRAD=G?" ;A$
20 IF A$="D" THEN 100
30 IF A$="R" THEN 200
40 GRAD :GOSUB 300:GOTO 40
100 DEGREE :GOSUB 300:GOTO 100
200 RADIAN :GOSUB 300:GOTO 200
300 INPUT "SIN ?";B
310 PRINT "SIN" ;B;"=";SIN B
320 RETURN
```

1 **IF** condition **THEN** statement
2 **IF** condition statement

Abbreviations: none for IF; T, TH., THE.

Purpose

The IF...THEN verb pair is used to execute or not to execute a statement depending on conditions at the time the program is run.

Use

In the normal running of BASIC programs, statements are executed in the sequence in which they occur. The IF...THEN verb pair allows decisions to be made during execution so that a given statement is executed only when desired. When the condition part of the IF statement is true, the statement is executed; when it is false, the statement is skipped.

The condition part of the IF statement can be any relational expression as described in Chapter 4. It is also possible to use a numeric expression as a condition, although the intent of the statement will be less clear. Any expression which evaluates to zero or a negative number is considered false; any which evaluates to a positive number is considered true.

The statement which follows the THEN may be any BASIC statement, including another IF...THEN. If it is a LET statement, the LET verb itself must appear.

The two formats of the IF statement are identical in action, but the first format is clear.

Note: The THEN verb before a statement cannot be omitted if the statement begins with any of the LLINE, RLINE, GLCURSOR, LTEXT, GRAPH, LF, CSIZE, COLOR, SORGN, CROTATE, CIRCLE, and PAINT verbs.

Examples

```
10 INPUT "CONTINUE?"; A$
20 IF A$="YES" THEN GOTO 10
30 IF A$="NO" THEN GOTO 60
40 PRINT "YES OR NO, PLEASE"
50 GOTO 10
60 END
```

This program continues to ask 'CONTINUE?' as long as 'YES' is entered; it stops if 'NO' is entered, and complains otherwise.

1 **INPUT** input list		
Where:	input list	is: input group or: input group, input list
and:	input group	is: var list or: prompt, var list or: prompt, var list
and:	var list	is: variable or: variable, var list
and:	prompt	is: any string constant

Abbreviations: I., IN., INP., INPU.

See also: INPUT#, READ, PRINT

Purpose

The INPUT verb is used to enter one or more values from the keyboard.

Use

When you want to enter different values each time a program is run, use the INPUT verb to enter these values from the keyboard.

In its simplest form the INPUT statement does not include a prompt string; instead a question mark is displayed at the left edge of the display. A value is then entered, followed by the [ENTER] key. This value is assigned to the first variable in the list. If other variables are included in the same INPUT statement, this process is repeated until the list is exhausted.

If a prompt is included in the INPUT statement, the process is exactly the same except that, instead of the question mark, the prompt string is displayed at the left edge of the display. If the prompt string is followed by a semicolon, the cursor is positioned immediately after the prompt. If the prompt is followed by a comma, the prompt is displayed. Then when a key is pressed, the display is cleared and the first character of the input is displayed at the left edge.

When a prompt is specified and there is more than one variable in the list following it, the second and succeeding variables are prompted with the question mark. If a second prompt is included in the list, it is displayed for the variable which immediately follows it.
If the [ENTER] key is pressed and no input is provided, the variable retains the value it had before the INPUT statement.

Examples

1Ø INPUT A	Clears the display and puts a question mark at the left edge.
2Ø INPUT "A=";A	Displays 'A=' and waits for input data.
3Ø INPUT "A=",A	Displays 'A='. When data is input, 'A=' disappears and the data is displayed starting at the left edge.
4Ø INPUT "X=?";X,"Y=?";Y	Displays 'X=?' and waits for first input. After [ENTER] is pressed, display is cleared and 'Y=?' is displayed at the left edge.

Note: Clear the error during input for the INPUT command by pressing the [C·CE] key and then input the correct data.

1 **INPUT** # var list
2 **INPUT** # "filename"; var list

Where: var list is: variable
or: variable, var list

Abbreviations: I. #, IN. #, INP. #, INPU. #

See also: INPUT, PRINT #, READ

Purpose

The INPUT # verb is used to enter values from the cassette tape.

Use and Examples

The following variable types can be specified in the INPUT # statement:
(1) Fixed variables–A, B, C, A(7), D*, A(20)*, etc.
(2) Simple variables–AA, B3, CP$, etc.
(3) Array variables–S(*), HP(*), K$(*), etc.

1) Transferring data to fixed variables

To transfer data from tape to fixed variables, specify the variable names in the INPUT # statement.

INPUT # "DATA 1" ; A, B, X, Y

This statement transfers data from the cassette file named "DATA1" to the variables A, B, X, and Y in that order.

To fill all the available fixed variables and, if defined, the extended variables (A(27) and beyond) with data transferred from tape, specify the first variable with an asterisk (*) subscripted to it.

INPUT # "D-2"; D*

This statement transfers the contents of the tape file "D-2" to variables D through Z and to A(27) and beyond.

INPUT # A(10)* (without DIM declaration)

This statement transfers the data of the first file found after the tape was started, to the variables A(10) and beyond (to J through Z and A(27) and beyond).

Notes: 1. If an array named A is already defined by the DIM statement, it is not possible to define subscripted fixed variables in the form of A( ).

2. Data transfer to fixed variables and extended variables (A(27) and beyond) will continue until the end of the source data file on the tape is reached, but if the computer's memory becomes full, an error (ERROR 6) results.

2) Data transfer to simple variables

Data in a tape file can be transferred to simple variables by specifying the desired simple variable names in the INPUT # statement.

```
INPUT # "DM-1"; AB, Y1, XY$
```

This statement transfers data from the tape file named "DM-1" to simple variables AB, Y1, and XY$.

Notes: 1. Numeric data must be transferred to numeric simple variables, and character data must be simple character variables. Cross-transfer is not allowed.

2. Locations for simple variables must be set aside in the program data area before the INPUT statement is executed. If not, an error will result. Use assignment statements to reserve the locations for simple variables.

   AA=Ø [ENTER]
   B1$="A" [ENTER]
   INPUT AA, B1$ [ENTER]

   Use appropriate numeric values or characters in assignment statements to reserve locations for variables.

3) Data transfer to array variables

To transfer data from a tape file to array variables, specify the array name in the INPUT # statement in the form of array name(*).

```
5Ø DIM B(5)
6Ø INPUT # "DS-4"; B(*)
```

This statement transfers data from the tape file named "DS-4" to the variables (B(0) through B(5)) in array B.

Note: 1. Numeric data must be transferred to numeric array variables with the same length as that of the data, character data must be transferred to character array variables with the same length as that of the data. If this rule is not observed, an error will result.

2. Locations for array variables must be set aside in the program data area before the INPUT # statement is executed. If not, an error will result. Use the DIM statement to define the array in advance.

–CAUTION–

If the number of variables specified in the INPUT statement does not agree with the amount of data recorded on the tape, the following will happen:

* If the number of pieces of data recorded on the tape file (to be transferred) is greater than the number of specified variables, data transfer will be performed to the last variable, and the remaining data will be ignored.
* If the number of pieces of data recorded in the tape file (to be transferred) is smaller than the number of specified variables, all the file data will be transferred to the variables to the end of the file, and the remaining variables will maintain their previous contents. In this case, however, the computer will continue to wait for data transfer from the tape. To halt this state, you should operate the ON BRK key.
* If the INPUT statement is executed with no variable name specified in it, an error (ERROR 1) will result.

1 **LET** variable=expression
2 variable=expression

Abbreviations: LE.

Purpose

The LET verb is used to assign a value to a variable.

Use

The LET verb assigns the value of the expression to the designated variable. The type of the expression must match that of the variable, i.e., only numeric expressions can be assigned to numeric variables and only string expressions can be assigned to string variables. To convert from one type to the other, one of the explicit type conversion functions, STR$ or VAL, must be used.

The LET verb may be omitted in all LET statements except those which appear in the THEN clause of an IF...THEN statement. In this one case the LET verb must be used.

Examples

```
1Ø I=1Ø                          Assigns the value '10' to I.
2Ø A=5*I                         Assigns the value '50' to A.
3Ø X$=STR$(A)                    Assigns the value '50' to X$
4Ø IF I>=1Ø THEN LET Y$=X$+".ØØ" Assigns the value '50.00' to Y$.
```

For printer CE-126P

1 **LPRINT** { expression / character string }

2 **LPRINT** { expression / character string } , { expression / character string }

3 **LPRINT** { expression / character string } ; { expression / character string } ;...; { expression / character string }

Abbreviations: LP., LPR., LPRI., LPRIN.

See also: PRINT, USING

Purpose

The LPRINT verb is used to print information on the printer CE-126P.

Use

When the serial I/O interface is opened with the OPEN command, the LPRINT command outputs the program at the serial I/O interface terminal. (See page 232.) To return the program printing command to the printer (CE-126P), execute the CLOSE command.

In format 1, numerics are right justified and alphabetics are printed from the left side of the paper. A line feed command is automatically executed when one line contains more than 24 characters.

In format 2, the 24 columns of one line are divided into two groups of 12 columns, and data is printed symmetrically around the comma.

A numeric value within the 12-column (digit) range is printed at the far right of the display, while a character value (string value) is printed starting at the far left. If the value to be printed exceeds 12 columns, the numeric value is printed with the least significant digit(s) of its decimal fraction part truncated so the value is within 12 digits, and the characters value is printed from the first 12 characters (from the left).

In format 3, the values are printed from the left edge of the paper. If the value to be printed exceeds 24 columns, a new line is automatically performed. Up to a maximum of 96 characters can be printed.

If both the CE-126P and CE-140P printers are connected, the CE-126P takes priority in printing over the CE-140P.

Note: Do not use any BASIC command or verb as a string expression.

Examples

```
10 A=10:B=20:X$="ABCDE":Y$="XYZ"
20 LPRINT A
30 LPRINT X$
40 LPRINT A,B
50 LPRINT X$;A;B
60 LPRINT
```

For printer CE-140P

1 **LPRINT** {expression / character string}

2 **LPRINT** {expression / character string}, {expression / character string},..., {expression / character string}

3 **LPRINT** {expression / character string}; {expression / character string};...; {expression / character string}

4 **LPRINT** ... {expression / character string};

(Format where a ";" is added to the end of 1 and 3 above.)

5 **LPRINT**

Abbreviations: LP., LPR., LPRI., LPRIN.

See also: PRINT, CONSOLE, USING

Purpose

The LPRINT verb is used to print information on the printer.

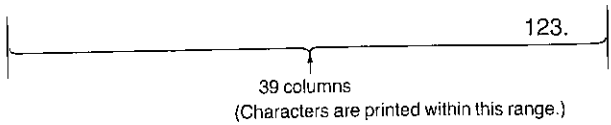
Use

When the serial I/O interface is opened with the OPEN command, the LPRINT command outputs the program at the serial I/O interface terminal. (See page 232.) To return the program printing command to CE-140P, execute the CLOSE command. (See page 225.)

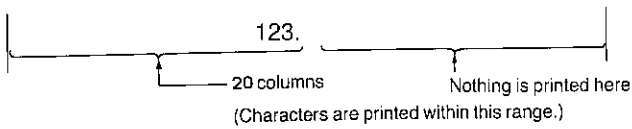
In format 1, the numerics are right justified and the characters are printed from the left end of the paper within the range of column specified in the CONSOLE command (usually 39 columns). (See page 226 for information on the CONSOLE command.)

Example: LPRINT 123 [ENTER] is executed with standard character size (size b)

1. When 39 columns are specified per printing line using the CONSOLE command



2. When 20 columns are specified per printing line using the CONSOLE command



In format 2, the number of print columns is delimited into groups of 12 columns. The specified values are printed in sequence. A numeric value within the 12-column (digit) range is printed at the far right end of the display, while a character value (string value) is printed starting at the far left. If the value to be printed exceeds 12 columns, the numeric value is printed with the least significant digit(s) of the decimal fraction truncated, so the value is within 12 digits, and the character value is printed from the first 12 characters (from the left).

The maximum number of values (items) (i.e., number of items which can be specified and separated with commas) specified in format 2 is 8. Specification of more items will result in ERROR 1.

In format 3, the values are printed from the left edge of the paper. If the value to be printed exceeds 24 columns, a new line is automatically performed. Up to a maximum of 96 characters can be printed.

In format 4, at the specified printing value, the value specified in the LPRINT command to be executed next will be printed in succession.

Note: If there is only one value to be printed in the equation printed by the LPRINT command following a command written in format 4, specify this value with ; " " following the equation.

Ex. 10 A=50:B=80
20 LPRINT "A*B=" ;
30 LPRINT A*B; "␣"

Use ; "␣" to specify results to be printed with line 30 following printing of line 20.

In format 5, no printing occurs but the paper is fed one line.

Note: Do not use any BASIC command or verb as a character string in a LPRINT statement.

Examples

```
10 A=10:B=20:X$="ABCDE":Y$="XYZ"
20 LPRINT A
30 LPRINT X$
40 LPRINT A,B,X$,Y$
50 LPRINT X$;A;B
60 LPRINT
70 LPRINT A*B;
80 LPRINT Y$
```

1 **MDF** expression

Abbreviation: MD.

See also: USING

Purpose

The MDF verb is used to round up the value of an expression.

Use

The MDF is a function used to round the value of an expression to the number of decimal places specified by the USING command.

This verb is effective only when the number of decimal places is specified for a value by the USING command.

Example

```
USING "###.###"
MDF (Ø.5/9)
```

Display: **Ø.Ø56**

```
1Ø USING "###.###"
2Ø A=MDF (5/9)
3Ø PRINT A
4Ø USING
5Ø PRINT A, 5/9
6Ø END
```

RUN [ENTER]

Ø.556

[ENTER]

Ø.556 5.55555E-Ø1

1 **NEXT** numeric variable

Abbreviations: N., NE., NEX.

See also: FOR

Purpose

The NEXT verb is used to mark the end of a group of statements which are being repeated in a FOR/NEXT loop.

Use

The use of the NEXT verb is described under FOR. The numeric variable in a NEXT statement must match the numeric variable in the corresponding FOR.

Examples

```
10 FOR I=1 TO 10
20 PRINT I
30 NEXT I
```

Print the numbers from 1 to 10 each time the ENTER is pressed.

1 **ON** expression **GOSUB** expression list
Where: expression list is: expression
or: expression, expression list

Abbreviations: O.; GOS., GOSU.

See also: GOSUB, GOTO, ON...GOTO

Purpose

The ON...GOSUB verb is used to execute one of a set of subroutines depending on the value of a control expression.

Use

When the ON...GOSUB verb is executed, the expression between ON and GOSUB is evaluated and reduced to an integer. If the value of the integer is 1, the first subroutine in the list is executed as in a normal GOSUB. If the expression is 2, the second subroutine in the list is executed, and so forth. After the RETURN from the subroutine, execution proceeds with the statement which follows the ON...GOSUB.

If the expression is zero, negative, or larger than the number of subroutines provided in the list, no subroutine is executed and execution proceeds with the next line of the program.

NOTE: Commas cannot be used in the expressions following the GOSUB. The **COMPUTER** cannot distinguish between commas in expressions and commas between expressions.

Examples

```
10 INPUT A
20 ON A GOSUB 100, 200, 300
30 END
100 PRINT "FIRST"
110 RETURN
200 PRINT "SECOND"
210 RETURN
300 PRINT "THIRD"
310 RETURN
```

An input of 1 prints "FIRST"; 2 prints "SECOND"; 3 prints "THIRD". Any other input does not produce any print.

1 **ON** expression **GOTO** expression list
Where: expression list is: expression
or: expression, expression list

Abbreviations: O.: G., GO., GOT.

See also: GOSUB, GOTO, ON...GOSUB

Purpose

The ON...GOTO verb is used to transfer control to one of a set of locations depending on the value of a control expression.

Use

When the ON...GOTO verb is executed, the expression between ON and GOTO is evaluated and reduced to an integer. If the value of the integer is 1, control is transferred to the first location in the list. If the expression is 2, control is transferred to the second location in the list, and so forth.

If the expression is zero, negative, or larger than the number of locations provided in the list, execution proceeds with the next line of the program.

NOTE: Commas cannot be used in the expressions following the GOTO. The **COMPUTER** cannot distinguish between commas in expressions and commas between expressions.

Examples

```
10 INPUT A
20 ON A GOTO 100, 200, 300
30 GOTO 900
100 PRINT "FIRST"
110 GOTO 900
200 PRINT "SECOND"
210 GOTO 900
300 PRINT "THIRD"
310 GOTO 900
900 END
```

An input of 1 prints 'FIRST'; 2 prints 'SECOND'; 3 prints 'THIRD'. Any other input does not produce any print.

1 **PAUSE** print expr
2 **PAUSE** print expr, print expr
3 **PAUSE** print expr; print list; ...; print list

Where: print list is: print expr
or: print expr; print list
and: print expr is: expression
or: USING clause; expression

The USING clause is described separately under USING.

Abbreviations: PAU., PAUS.

See also: LPRINT, PRINT, USING, WAIT

Purpose

The PAUSE verb is used to print information on the display for a short period.

Use

The PAUSE verb is used to display prompting information, results of calculations, etc. The operation of PAUSE is identical to PRINT except that after PAUSE the **COMPUTER** waits for a short preset interval of about 0.85 second and then continues execution of the program without waiting for the ENTER key or the WAIT interval.

The first form of the PAUSE statement displays a single value. If the expression is numeric, the value is printed at the far right of the display. If it is a string expression, the display is made starting at the far left.

In format 2, the display unit is divided into groups of 12 columns. The values are displayed, in sequence, from the first specified value.

In this case too, within a range of 12 columns, the numeric value of an expression is displayed from the right end of the display and characters are displayed from the left side.

- The number of the values (items) specified in format 2 must be within 2.
- If the specified value exceeds 12 columns, the following is performed.

1) When the numeric value exceeds 12 digits, the least significant digit(s) is truncated.
2) When the characters exceed 12 columns, only the first 12 characters (from the left) are displayed.

In format 3, the specified value is displayed continuously from the left side of the display.

Examples

```
10 A=10: B=20: X$="ABCDEF":
Y$="XYZ"
```

	Display
20 PAUSE A	**10.**
30 PAUSE X$	**ABCDEF**
40 PAUSE X$,B	**ABCDEF** **20.**
50 PAUSE Y$;X$	**XYZABCDEF**
60 PAUSE A*B	**200.**

Note: Do not use any BASIC command or verb as a character string in a PAUSE statement.

1 **PRINT** print expr
2 **PRINT** print expr, print expr
3 **PRINT** print list
4 **PRINT=LPRINT**
5 **PRINT=PRINT**

Where: print list is: print exp
or: print expr; print list
and: print expr is: expression
or: USING clause; expression

The USING clause is described separately under USING.

Abbreviations: P., PR., PRI., PRIN.

See also: LPRINT, USING, WAIT

Purpose

The PRINT verb is used to print information on the display or on the printer.

Use

The PRINT verb is used to display prompting information, results of calculations, etc. The first form of the PRINT statement displays a single value. If the expression is numeric, the value is printed at the far right of the display. If it is a string expression, the display is made starting at the far left.

In format 2, the display unit is divided into groups of 12 columns. The values are displayed, in sequence, from the first specified value. In this case too, within a range of 12 columns, the value of an expression is displayed from the right end of the display and characters are displayed from the left side.

- The number of the values (items) specified in format 2 must be within 2.
- If the specified value exceeds 12 columns, the following is performed.

1) When the numeric value exceeds 12 digits, the least significant digit(s) is truncated.
2) When the characters exceed 12 columns, only the first 12 characters (from the left) are displayed.

In format 3, the specified value is displayed continuously from the left side of the display.

Note: Do not use any BASIC command or verb as a character string in a PRINT statement.

Examples

```
10 A=123:B=456:X$="ABCDEF":
   Y$="VWXYZ"
20 PRINT X$,B
```

Display

```
ABCDEF          456.
```

```
30 PRINT A;B
```

```
123.456.
```

```
40 PRINT X$;A
```

```
ABCDEF123.
```

```
50 PRINT Y$;B
```

```
VWXYZ456.
```

1 **PRINT** # "var list"
2 **PRINT** # "filename" ; var list

Where: var list is: variable
or: variable, var list

Abbreviations: P. #, PR. #, PRI. #,PRIN. #

See also: INPUT #, PRINT, READ

Purpose

The PRINT # verb is used to store values on the cassette tape.

Use and Examples

The following variable types can be used for variable names:
(1) Fixed variables—A, B, X, A(26), C*, A(1Ø)*, etc.
(2) Simple variables—AA, B2, XY$, etc.
(3) Array variables—B(*), CD(*), NS(*), etc.

1) Saving fixed variable contents onto tape

The contents of fixed variables can be saved onto tape by specifying the desired variable names (separated by commas) in the PRINT # statement.

```
PRINT # "DATA 1" ; A, B, X, Y
```

This statement saves contents of variables A, B, X, and Y into tape file named "DATA 1".
If you wish to save the contents of the specified fixed variable and all the subsequent fixed variables, subscript that variable name with an asterisk*.

PRINT # "D-2"; D*	This statement saves the contents of fixed variables D through Z (and of extended variables A(27) and beyond, if defined) into the tape file named "D-2".
PRINT E,X$,A(3Ø)*	This statement saves the contents of the fixed variables E and X$ and of the extended variables A(3Ø) and all the remaining variables, onto the tape without file name.

Note: Subscripted fixed variable names A(1) through A(26) can be specified in the PRINT # statement in much the same way as A through Z (or A$ through Z$). However, if array A is already defined by the DIM statement, A( ) cannot be used to define subscripted fixed variables.

2) Saving simple variable (two-character variable) contents

The contents of simple variables can be saved onto tape by specifying the desired variable names.

PRINT # "DM-1"; AB, Y1, XY$

This statement saves the contents of the simple variables AB, Y1, and XY$ into the tape file named 'DM-1'.

3) Saving array variable contents

The contents of all variables of a specific array can be saved onto tape by specifying the array name subscripted by an asterisk enclosed in parentheses (*).

PRINT # "DS-2";X(*),YS(*)

This statement saves the contents of all the elements (X(0),X(1), ...) of the array X, and of all the elements (X$(0), Y$(1), ...) of the array Y$, into the tape file name 'DS-2'.

Note: It is not possible to save the contents of only one specific element of an array. While fixed variables or subscripted fixed variables in the form of A( ) allow you to save only one specific element of such a variable, an array (such as A), defined by the DIM statement, allows you to save in the same manner as other arrays.

* If the PRINT # statement is executed with no variable names specified, an error (ERROR 1) will result.

–CAUTION–

The locations for extended variables such as A(27) and beyond, simple variables, and/or array variables must be set aside in the program/data area before the PRINT # statement is executed. Otherwise, the execution of the PRINT # statement for undefined variables will result in an error.

1 **RADIAN**

Abbreviations: RAD., RADI., RADIA.

See also: DEGREE, GRAD

Purpose

The RADIAN verb is used to change the unit of angle to radians.

Use

The **COMPUTER** has three forms for representing values in angular values—decimal degrees, radians, and grads. These forms are used in specifying the arguments to the SIN, COS, and TAN functions and in returning the results from the ASN, ACS, and ATN functions.

The RADIAN function changes the unit of angle for all values to radians until a DEGREE or GRAD verb is used. Radian represents an angular measurement in terms of the length of the arc with respect to a radius, i.e., 360° is 2 PI radians since the circumference of a circle is 2 PI times the radius.

Examples

```
1Ø RADIAN
2Ø X=ASN 1
3Ø PRINT X
```

X now has a value of 1.570796327 or PI/2, the arc sine of 1.

1 **RANDOM**

Abbreviations: RA., RAN., RAND., RANDO.

Purpose

The RANDOM verb is used to reset the seed for random number generation.

Use

When random numbers are generated using the RND function, the **COMPUTER** begins with a predetermined "seed" or starting number. The RANDOM verb resets this seed to a new randomly determined value.

The starting seed will be the same each time the **COMPUTER** is turned on, so the sequence of random numbers generated with RND is the same each time, unless the seed is changed. This is very convenient during the development of a program because it means that the behavior of the program should be the same each time it is run, even though it includes an RND function. When you want to have the numbers be truly random, the RANDOM statement can be used to make the seed itself random.

Examples

```
1Ø RANDOM
2Ø X=RND 1Ø
```

When run from line 20, the value of X is based on the standard seed. When run from line 10, a new seed is used.

1 **READ** variable list
Where: variable list is: variable
or: variable , variable list

Abbreviations: REA.

See also: DATA, RESTORE

Purpose

The READ verb is used to read values from a DATA statement and assign them to variables.

Use

When assigning initial values to an array, it is convenient to list the values in a DATA statement and use a READ statement in a FOR...NEXT loop to load the values into the array. When the first READ is executed, the first value in the first DATA statement is returned. Succeeding READs use succeeding values in the sequential order in which they appear in the program, regardless of how many values are listed in each DATA statement or how many DATA statements are used.

If desired, the values in a DATA statement can be read a second time by using the RESTORE statement.

Examples

```
10 DIM B (10)
20 WAIT 32
30 FOR I=1 TO 10
40 READ B(I)
50 PRINT B(I)*2
60 NEXT I
70 DATA 10 20 30 40 50 60
80 DATA 70 80 90 100
90 END
```

Set up an array.

Loads the values from the DATA statement into B( )—
B(1) is 10, B(2) is 20, B(3) is 30, etc.

1 **REM** remark

Abbreviations: none

Purpose

The REM verb is used to include comments in a program.

Use

Often it is useful to include explanatory comments in a program. These can provide titles, names of authors, dates of last modification, usage notes, reminders about algorithms used, etc. These comments are included by means of the REM statement.

The REM statement has no effect on the program execution and can be included anywhere in the program. Everything following the REM verb in that line is treated as a comment.

Example

```
1Ø REM THIS LINE HAS NO EFFECT
```

1 **RESTORE**
2 **RESTORE** expression

Abbreviations: RES., REST., RESTO., RESTOR.

See also: DATA, READ

Purpose

The RESTORE verb is used to reread values in a DATA statement or to change the order in which these values are read.

Use

In the regular use of the READ verb, the **COMPUTER** begins reading with the first value in a DATA statement and proceeds sequentially through the remaining values. The first form of the RESTORE statement resets the pointer to the first value of the first DATA statement, so that it can be read again. The second form of the RESTORE statement resets the pointer to the first value of the first DATA statement whose line number is greater than the value of the expression.

Examples

```
1Ø DIM B(1Ø)
2Ø WAIT 32
3Ø FOR I=1 TO 1Ø
4Ø RESTORE
5Ø READ B(I)
6Ø PRINT B(I)*I
7Ø NEXT I
8Ø DATA 2Ø
9Ø END
```

Sets up an array.

Assign the value 20 to each of the elements of B( ).

1 RETURN

Abbreviations: RE., RET., RETU., RETUR.

See also: GOSUB, ON...GOSUB

Purpose

The RETURN verb is used at the end of a subroutine to return control to the statement following the originating GOSUB.

Use

A subroutine may have more than one RETURN statement, but the first one executed terminates the execution of the subroutine. The next statement executed will be the one following the GOSUB or ON...GOSUB which calls the subroutine. If a RETURN is executed without a GOSUB, an ERROR 5 will occur.

Examples

```
10 GOSUB 100
20 END
100 PRINT "HELLO"
110 RETURN
```

When run, this program prints the word "HELLO" one time.

1 **STOP**

Abbreviations: S., ST., STO.

See also: END, CONT

Purpose

The STOP verb is used to halt execution of a program for diagnostic purposes.

Use

When the STOP verb is encountered in program execution, the **COMPUTER** execution halts and a message is displayed such as 'BREAK IN 2ØØ' where 2ØØ is the number of the line containing the STOP. STOP is used during the development of a program to check the flow of the program or examine the state of variables. Execution may be restarted using the CONT command.

Example

```
1Ø STOP
```

Causes "BREAK IN 1Ø" to appear in the display.

1 **TROFF**

Abbreviations: TROF.

See also: TRON

Purpose

The TROFF verb is used to cancel the trace mode.

Use

Execution of the TROFF verb restores normal execution of the program.

Examples

```
1Ø TRON
2Ø FOR I=1 TO 3
3Ø NEXT I
4Ø TROFF
```

When run, this program displays the line numbers 10, 20, 30, 30, 30, and 40 as the [↓] is pressed.

1 **TRON**

Abbreviations: TR., TRO.

See also: TROFF

Purpose

The TRON verb is used to initiate the trace mode.

Use

The trace mode provides assistance in debugging programs. When the trace mode is on, line number of each statement is displayed after each statement is executed. The **COMPUTER** then halts and waits for the Down Arrow key to be pressed before moving on to the next statement. The Up Arrow key may be pressed to see the statement which has just been executed. The trace mode continues until a TROFF verb is executed or the key operation of the [SHIFT] and [CA C·CE] is performed.

Examples

```
10 TRON
20 FOR I=1 TO 3
30 NEXT I
40 TROFF
```

When run, this program displays the line numbers 10, 20, 30, 30, 30, and 40 as the [↓] is pressed.

1 **USING**
2 **USING** "editing specification"

Abbreviations: U., US., USI., USIN.

See also: LPRINT, PAUSE, PRINT

Purpose

The USING verb is used to control the format of displayed or printed output.

Use

The USING verb can be used by itself or as a clause within an LPRINT, PAUSE, or PRINT statement. The USING verb establishes a specified format for output which is used for all output which follows until changed by another USING verb.

The editing specification of the USING verb consists of a quoted string composed of some combination of the following editing characters:

- # Right justified numeric field character
- . Decimal point
- ^ Used to indicate that numbers should be displayed in scientific notation
- & Left justified alphanumeric field

For example, "####" is an editing specification for a right justified numeric field with room for 3 digits and the sign. In numeric fields, a location must be included for the sign, even if it will always be positive.

Editing specifications may include more than one field. For example "####&&&&" could be used to print a numeric and a character field next to each other.

If the editing specification is missing, as in format 1, special formatting is turned off and the built-in display rules pertain.

Examples

Display

```
10 A=125:X$="ABCDEF"
20 PRINT USING "##.##^";A          1.25E 02
30 PRINT USING "&&&&&&&&";X$       ABCDEF
40 PRINT USING "####&&&";A;X$      125ABC
```

(See APPENDIX C for further guide to the use of USING.)

1 **WAIT** expression
2 **WAIT**

Abbreviations: W., WA., WAI.

See also: PRINT

Purpose

The WAIT verb is used to control the length of time that displayed information is shown before program execution continues.

Use

In normal execution, the **COMPUTER** halts execution after a PRINT command until the [ENTER] key is pressed. The WAIT command causes the **COMPUTER** to display for a specified interval and then proceed automatically (similar to the PAUSE verb). The expression which follows the WAIT verb determines the length of the interval. The interval may be set to any value from 0 to 65535. Each increment is about one fifty-ninth of a second. WAIT 0 is too fast to be read reasonably; WAIT 65535 is about 19 minutes. WAIT with no following expression resets the **COMPUTER** to the original condition of waiting until the [ENTER] key is pressed.

Example

```
1Ø WAIT 59
```

Causes PRINT to wait about 1 second.

Graphics Related Verbs

The following verbs are related to the graphic functions of the CE-140P color dot printer and CE-515P color plotter/ printer. Printing of data in graphic form can be performed by the method described in the operation manual of each printer (namely, transfer of graphics codes by LPRINT command). However, with these verbs, you can draw a graph or diagram much easier.
All the values of graphic related verbs except the ratio specification with CIRCLE are truncated to whole numbers when these verbs are executed.

1 **CIRCLE** (expression 1, expression 2), expression 3
[,expression 4][,expression 5][,expression 6]
[,expression 7][,expression 8][,expression 9]

Abbreviations: CI., CIR., CIRC., CIRCL.

See also: PAINT, COLOR

Purpose

The CIRCLE verb is used to draw a circle.

Use

This verb is effective only in the Graphics mode and is used to draw a circle, arc, sector, or ellipse in solid line.

Expressions 1 and 2 are used to specify coordinates X and Y, respectively, at the center point of a circle. The values of expressions 1 and 2 must be within the range of −999 to 999.

Expression 3 is used to specify the radius of a circle. The value of expression 3 must be within the range of 1 to 499 ($1 \leqq$ expression $3 \leqq 499$).

Expression 4 is used to specify the color of line. The value of expression 4 may be specified within the range of 0 to 7 in the extended Color mode and 0 to 3 in other than the extended Color mode. (See the COLOR verb for the colors specified by the respective values.)
If expression 4 is omitted, the previous value (i.e., the color previously specified) is assumed.

Expressions 5 and 6 are used to specify the starting angle and ending angle, respectively, of an arc or sector. The respective values of expressions 5 and 6 must be within the range of −2047 to 2047. If the value is 0, the right side of the central coordinates is specified. If a negative value is given, the counterclockwise direction is specified. If a positive value is given, the clockwise direction is specified.
The default value of expression 5 is 0 degree, and that of expression 6 is 360 degrees.

Expression 7 is used to specify the following ratio.

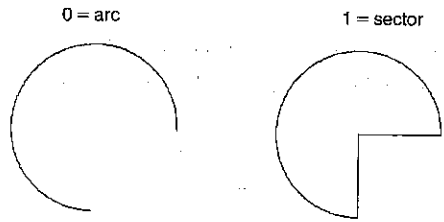
$$\text{Ratio} = \frac{r_y \text{ (radius in Y-axis direction)}}{r_x \text{ (radius in X-axis direction)}}$$

If the value of expression 7 is 1, a circle is drawn. If the given value is other than 1, an ellipse is drawn. The default value of expression 7 is 1.

Expression 8 is used to specify a pitch angle. The printer starts drawing a circle (arc or sector) or ellipse by dividing it in units of pitches from the starting angle to the ending angle. The value of expression 8 must be within the range of 1 to 2048. The default value of expression 8 is 1.

Expression 9 is used to specify an arc or a sector. If the value of expression 9 is 0, an arc is drawn. If the value given is 1, a sector is drawn. The default value of expression 9 is 0.

Note: The respective values of expressions 5, 6, and 8 are specified in units of degrees irrespective of the specified angular mode.



Examples

```
5 OPEN
1Ø GRAPH
2Ø CIRCLE (24Ø, −1ØØ), 1ØØ, Ø, Ø,
   36Ø, 1/2, 1Ø, Ø
3Ø LTEXT
4Ø LPRINT
5Ø END
```

This verb is required for CE-515P.

Ratio = 0.5

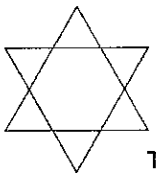
These two verbs return the printer to the Text mode and move the print head back to its leftmost position.



```
5 OPEN
1Ø GRAPH
2Ø CIRCLE (24Ø, −1ØØ), 1ØØ, Ø,
   9Ø, 45Ø, 1, 12Ø, Ø
3Ø CIRCLE (24Ø, −1ØØ), 1ØØ, 3,
   −9Ø, 27Ø, 1, 12Ø, Ø
4Ø LTEXT
5Ø LPRINT
6Ø END
```

Pitch angle = 120

Pitch angle = 120



The figure is actually printed in color.

1 **COLOR** expression
2 **COLOR** expression, 7

Abbreviations: COL., COLO.

See also: LLINE, RLINE, CIRCLE, PAINT

Purpose

The COLOR verb is used to specify the color of characters or lines to be printed in normal or extended Color mode.

Use

Using format 1 (COLOR expression), you can specify four different colors by giving 0 to 3 as the value of expression.

Using format 2 (COLOR expression, 7), you can specify eight different colors by giving 0 to 7 as the value of expression.

With the CE-140P color dot printer, both the formats can be used to specify colors. However, the CE-515P color plotter/printer can only use format 1. The color specifications applicable to the respective printers are as follows:

CE-140P

Value of expression	0	1	2	3	4	5	6	7
Format 1	Black	Purple	Green	Red				
Format 2	Black	Purple	Red	Magenta	Green	Cyan	Yellow	White

Note: The white specified by value "7" refers to the color of a printing paper. When this value is given, both the print head and the printing paper move, but no printing is executed.

CE-515P

Value of expression	0	1	2	3
Format 1	Black	Blue	Green	Red

When the COLOR verb is executed with format 2, the printer is automatically put in the extended Color mode. In this mode, you can select a desired color from the abovementioned eight colors when using the LLINE, RLINE, CIRCLE, or PAINT verb.
The printer is released from the extended Color mode when the power switch of the computer is turned off.

Example

```
1Ø GRAPH
2Ø COLOR 1, 7
```

Sets the extended Color mode and at the same time, specifies "Purple" as the color to be used. (CE-140P must be used in this case.)

1 **CROTATE** expression

Abbreviations: CR., CRO., CROT., CROTA., CROTAT.

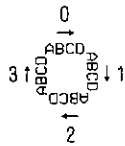
Purpose

The CROTATE verb is used to specify the orientation and printing direction of characters to be printed.

Use

This verb is effective only when the printer is in the Graphic mode. By changing the value of expression, you may change the printing direction and orientation of characters to be printed.

If you give a value in the range of 0 to 3 to the expression, one of the following four orientations and printing directions of characters is specified, causing the printer to perform printing in the direction indicated by the arrow.



When the LTEXT verb is executed, the orientation and printing direction of characters to be printed change automatically to normal as when the value 0 is given.

Example

```
5 OPEN
1Ø GRAPH
2Ø GLCURSOR (2ØØ, −3Ø)
3Ø FOR Z=Ø TO 3
4Ø CROTATE Z
5Ø LPRINT "PABCD"
6Ø NEXT Z
7Ø LTEXT
8Ø LPRINT
9Ø END
```

This command is required for the CE-515P.

1 **CSIZE** expression

Abbreviations: CSI., CSIZ.

Purpose

The CSIZE verb is used to specify the size of characters to be printed.

Use

The size of characters to be printed can be specified by giving a value within the following range to the expression.

CE-140P	CE-515P
1 to 63	1 to 15

Assuming that the value of expression given as "1" is the minimum size of characters, the size of characters specified by 2, 3, 4, ... will be two, three, four, ... times the minimum character size as well as the character pitch and line spacing.

Example

CSIZE 1

Character size : 0.8 mm(W) × 1.2 mm(H) (4 × 6 steps)
Character pitch: 1.2 mm (6 steps)
Line spacing : 2.4 mm (12 steps)

Note: When the power switch of the printer is turned on, the printer is in the state to print characters by CSIZE 2.

1 **GLCURSOR** (expression 1, expression 2)

Abbreviations: GL., GLC., GLCU., GLCUR., GLCURS., GLCURSO.

Purpose

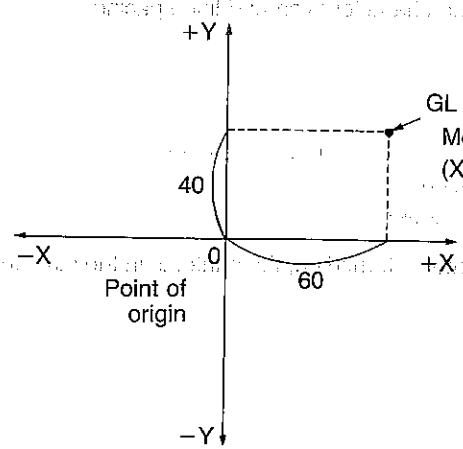
The GLCURSOR verb is used to move the pen.

Use

This verb is effective only in the Graphics mode and is used to move the pen in the X-axis or Y-axis direction from the origin of coordinates.

The pen moves to the point of coordinates specified by expression 1 (X coordinate) and expression 2 (Y coordinate).

The value of each expression must be within the range of −999 to 999. The minimum amount of movement is 0.2 mm in either direction.



Note: Moving the pen in the +Y direction means that the paper feeding must be effected in the reverse direction.
Refer to the operation manual of the printer for the scissoring area.

Example

```
10 GRAPH
20 GLCURSOR (60,40)
```

Moves the pen to the point of coordinates (X=60, Y=40).

1 **GRAPH**

Abbreviation: GRAP.

See also: LTEXT

Purpose

The GRAPH verb is used to set the printer in the Graphics mode.

Use

When this verb is executed, the printer is released from the Text mode and put in the Graphics mode for drawing a graph.

The printer is automatically released from the Graphics mode and returns to the Text mode after the execution of the LLIST command or after printing by the manual operation with the CE-140P connected.

To print characters in the Graphics mode, the LPRINT verb is used in either of the following two formats:

(1) LPRINT "P character string"

(2) LPRINT "P" {,/;} string variable

Notes: 1. If you interrupt the printer operation by pressing the [BRK] key while the printer is drawing a figure in the Graphics mode, be sure to enter:

LPRINT "" [ENTER]

If you fail to do this, subsequent commands to the printer may not be executed properly.

2. To return the print head to its leftmost position, operate:

LTEXT [ENTER]

LPRINT "" [ENTER]

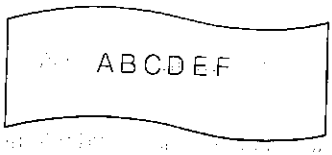
In this case, however, the printer will be released from the Graphics mode.

3. When any of the printer-related commands or verbs effective only in the Graphics mode (CIRCLE, CROTATE, GLCURSOR, etc.) is used in the Text mode, the printer will print characters according to the statement character.

Example

```
 5 OPEN
1Ø GRAPH
2Ø GLCURSOR (2ØØ, −3Ø)
3Ø LPRINT "PABCDEF"
4Ø LTEXT
5Ø LPRINT
6Ø END
```

This command is required for the CE-515P.



1 **LF**
2 **LF** expression

Purpose

The LF verb is used to feed the printing paper.

Use

This verb is effective only in the Text mode.

With format 1, the printer feeds the paper by one line.

With format 2, the printer feeds the paper by the specified number of lines. The value of the expression must be within the range of −999 to 999.

If the value of the expression is a positive value, the paper is fed in the forward direction. If a negative value is given, the paper is fed in the reverse direction.

Note: The line spacing when the LF verb is executed will be the same as that specified by the CSIZE verb.

Example

```
LF 1Ø          Feeds the paper by 10 lines.
```

1 **LLINE** [(expression 1, expression 2)]–(expression 3, expression 4) [,expression 5][,expression 6][,B]

Abbreviation: LLIN.

See also: RLINE, PAINT, COLOR

Purpose

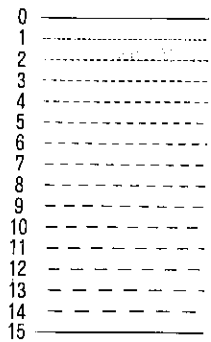
The LLINE verb is used to draw a line between two specified points.

Use

This verb is effective only in the Graphics mode and is used to draw a line from the point of coordinates specified by (expression 1, expression 2) to the point of coordinates specified by (expression 3, expression 4).

(expression 1, expression 2) may be omitted. If omitted, a line is drawn from the current position of the pen to the point specified by (expression 3, expression 4).

Expression 5 is used to specify one of the following types of lines by giving a value (0 to 15) to the expression. The default value of expression 5 is 0.



Expression 6 is used to specify the color of the line to be drawn. The value of expression 6 may be within the range of 0 to 7 in the extended Color mode and 0 to 3 in the normal Color mode. (Refer to the COLOR verb for the color specified by each value.)

Expression 6 may be omitted. If omitted, the previous value is assumed.

Example

```
 5 OPEN
```

This command is required for the CE-515P.

```
10 GRAPH : RANDOM
20 GLCURSOR (240, -120)
25 SORGN
```

Moves the pen to about the center of the paper and designates it as the origin of coordinates for drawing a figure.

```
30 FOR J=0 TO 340 STEP 20
40 A=107*COS J
50 B=107*SIN J
60 R=RND 4-1
```

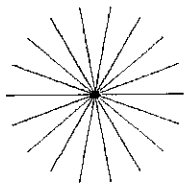
Random number 0 to 3 is assigned to R.

```
70 LLINE (0, 0)-(A,B), 0, R
80 NEXT J
90 LTEXT
```

Color is specified by the value of R.

The printer returns to the Text mode and moves the print head to its leftmost position.

```
100 LPRINT
110 END
```



This figure is actually printed in color.

If B is specified at the end of the LLINE verb, a rectangle is drawn using the point of coordinates specified by (expression 1, expression 2) and the point of coordinates specified by (expression 3, expression 4) as the end points of a diagonal line.

Example: LLINE (10,20) – (200, -20), 2, 0, B

Note: (expression 1, expression 2) cannot be omitted when drawing a rectangle.

Verb Specification When Drawing Lines Continuously

The LLINE verb in the following format allows the printer to draw lines continuously.

LLINE (expression 1, expression 2) – (expression 3, expression 4)
– (expression 5, expression 6) – (expression 7, expression 8)
– (expression 9, expression 10) – (expression 11, expression 12),
expression, expression

With the above format, a line connecting the point specified by (expression 1, expression 2) to the point specified by (expression 3, expression 4), a line connecting the point specified by (expression 3, expression 4), to the point specified by (expression 5, expression 6), ... can be drawn continuously. A maximum of six expression pairs can be specified.

1 **LTEXT**

Abbreviations: LT., LTE., LTEX.

See also: GRAPH

Purpose

The LTEXT verb is used to set the Text mode.

Use

This verb is used to put the printer in the Text mode for printing alphabetic and numeric characters.

Note: The printer is automatically put in the Text mode after the execution of the LLIST verb or after the printing by the manual operation with the CE-140P connected. Be sure to set the operation mode with either the GRAPH or LTEXT verb when the BRK key is pressed after the automatic power off of the computer or when the power switch of either the computer or the printer is turned on again from the OFF position.

Example

```
LTEXT     Sets the Text mode.
```

1 **PAINT** expression 1[,expression 2]

Abbreviations: PAI., PAIN.

See also: LLINE, RLINE, CIRCLE, COLOR

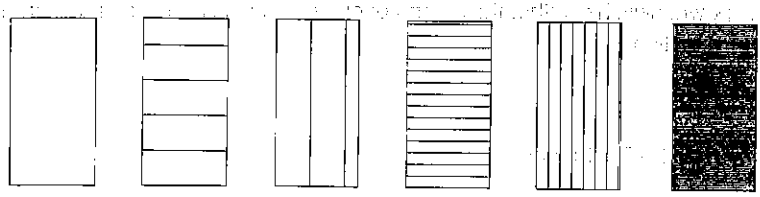
Purpose

The PAINT verb is used to hatch the inside of a rectangle or circle (or sector).

Use

This verb is effective only in the Graphics mode. When a rectangle or circle (or sector) has been drawn by the verb executed immediately before the PAINT verb, the printer hatches the inside of the rectangle or circle (or sector).

The following types of patterns can be specified for the rectangle drawn by the "B" specification of the LLINE or RLINE verb and for the circle (or sector) drawn by the CIRCLE verb by giving value (0, 1, 2, 5, 6, 9, or 10) to expression 1.





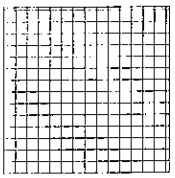
Repeated use of the PAINT verb together with the LLINE verb allows the printer to draw a special pattern as shown in Example.

Expression 2 is used to specify the color of hatching. The value of expression 2 must be within the range of 0 to 7 in the extended Color mode and 0 to 3 in the normal Color mode. (Refer to the COLOR verb for the color specified by each value.)

Note: To execute the PAINT verb for the circle (or sector) drawn by the CIRCLE verb, ratio = 1 and pitch angle = 1 must have been specified as the values of expressions 7 and 8 of the CIRCLE verb.

Example

```
LLINE (Ø, Ø) – (2ØØ, –2ØØ), Ø, Ø, B
PAINT 5, Ø
PAINT 6, Ø
```



1 **RLINE** (expression 1, expression 2) – (expression 3, expression 4)
[,expression 5][,expression 6][,B]

Abbreviations: RL., RLI., RLIN.

See also: LLINE, PAINT, COLOR

Purpose

The RLINE verb is used to draw a line between the two points specified by relative coordinates.

Use

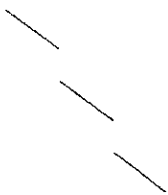
This verb is effective only in the Graphics mode.

- The RLINE verb differs from the LLINE verb in that the LLINE verb takes the origin of coordinates specified by the SORGN verb as a reference and specifies the location of each point by coordinates from that origin, whereas the RLINE verb takes the current position of the pen as the origin of coordinates and specifies the location of the next point by coordinates from that origin.

Descriptions of the verb specification are the same as the LLINE verb, except that (expression 1, expression 2) cannot be omitted. (See page 206.)

Example

```
5 OPEN
1Ø GRAPH
2Ø FOR A=1 TO 3
3Ø RLINE (Ø,-4Ø) - (6Ø,-5Ø)
4Ø NEXT A
5Ø LTEXT
6Ø LPRINT
7Ø END
```



1 **SORGN**

Abbreviations: SO., SOR., SORG.

Purpose

The SORGN verb is used to change the origin of coordinates for drawing with the pen.

Use

This verb is effective only in the Graphics mode and is used to specify the current position of the pen as the new origin of coordinates.
When drawing a figure, it may not be easier for the printer to do so if the origin of coordinates is located at the left end of the paper. In such a case, move the pen to an arbitrary position on the paper by executing the GLCURSOR verb and then specify that position as the origin of coordinates with the SORGN verb.
This would facilitate drawing of a figure with the current position of the pen taken as a reference point.

Example

```
1Ø GRAPH
2Ø GLCURSOR (6Ø, 4Ø)
3Ø SORGN
```

Specifies the current position (X=60, Y=40) of the pen as the new origin of coordinates.

FUNCTIONS

Pseudovariables

Pseudovariables are a group of functions which take no argument and are used like simple variables wherever required.

1 INKEY$

INKEY$ is a string pseudovariable which has the value of the last key pressed on the keyboard. [ENTER], [C-CE], [SHIFT], [DEF], [SML], [↑], [↓], [►], [◄], [CAL], [BASIC], and scientific function keys all have a value of null. INKEY$ is used to respond to the pressing of individual keys without waiting for the ENTER key to end the input.

```
5 WAIT 5Ø
1Ø A$=INKEY$
2Ø B=ASC A$
3Ø IF B=Ø THEN GOTO 1Ø
4Ø IF B ...
```

Lines 40 and beyond contain tests for the key and the actions to be taken (for example: 4Ø PRINT A$). On first executing the program, the value of INKEY$ is null, since the last key pressed was [ENTER].

- If an INKEY$ command is written at the beginning of the program, the start key may be read (by the INKEY$ command) when the program is started. For example, in the following program

  ```
  1Ø"Z" : Z$=INKEY$
  . . .
  ```

  The [Z] key may be read when the program is started by pressing the [DEF] [Z] keys.

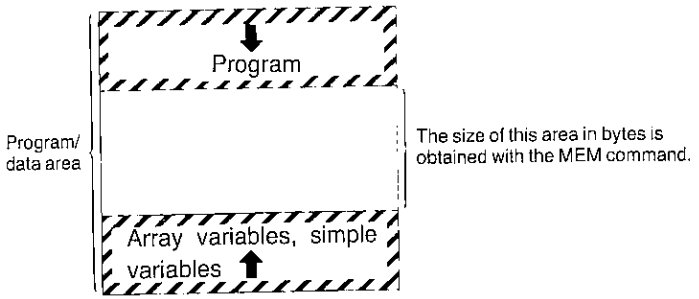
1 **MEM**
Abbreviations: M., ME.

Purpose

To obtain the number of free bytes in the program/data area.

Use

Obtains the number of free bytes (area not used by a program, array variables, or simple variables) in the program/data area.



Reference

The program size (in bytes) can be obtained by the following operation.

Example: When using RAM card CE-212M
RUN mode

CLEAR [ENTER] (Clears the simple variables, array variables, etc.)
6878-MEM [ENTER] ← displays the number of bytes in the program
This value varies depending on the use of the RAM card.

(See page 117 for the program/data area capacity within optional RAM card.)

1 PI

PI is a numeric pseudovariable which has the value of PI. It is identical to the use of the special PI character (π) on the keyboard. Like other numbers, the value of PI is kept to 10-digit accuracy (3.141592654).

Numeric Functions

Numeric functions are a group of mathematical operations which take a single numeric value and return a numeric value. They include trigonometric functions, logarithmic functions, and functions which operate on the integer and sign parts of a number. Many dialects of BASIC require that the argument to a function be enclosed in parentheses. The **COMPUTER** does not require these parentheses, except when it is necessary to indicate what part of a more complex expression is to be included in the argument.

LOG 1ØØ+1ØØ will be interpreted as:
(LOG 1ØØ)+1ØØ not LOG (1ØØ+1ØØ)

1 **ABS** numeric expression

ABS is a numeric function which returns the absolute value of the numeric argument. The absolute value is the value of a number without regard to its sign. ABS – 10 is 10.

1 **ACS** numeric expression

ACS is a numeric function which returns the arc cosine of the numeric argument. The arc cosine is the angle whose cosine is equal to the expression. The value returned depends on whether the **COMPUTER** is in decimal degree, radian, or grad mode for angles. ACS.5 is 60 in the decimal degree mode.

1 **AHC** numeric expression

AHC is a numeric function which returns arc-hyperbolic cosine of the numeric argument. AHC 5 is 2.29243167.

1 **AHS** numeric expression

AHS is a numeric function which returns arc-hyperbolic sine of the numeric argument. AHS 6 is 2.491779853.

1 **AHT** numeric expression

AHT is a numeric function which returns arc-hyperbolic tangent of the numeric argument.

1 **ASN** numeric expression

ASN is a numeric function which returns the arc sine of the numeric argument. The arc sine is the angle whose sine is equal to the expression. The value returned depends on whether the **COMPUTER** is in decimal degree, radian, or grad mode for angles. ASN.5 is 30 in the decimal degree mode.

1 **ATN** numeric expression

ATN is a numeric function which returns the arc tangent of the numeric argument. The arc tangent is the angle whose tangent is equal to the expression. The value returned depends on whether the **COMPUTER** is in decimal degree, radian, or grad mode for angles. ATN 1. is 45 in the decimal degree mode.

1 **COS** numeric expression

COS is a numeric function which returns cosine of the angle argument. The value returned depends on whether the **COMPUTER** is in decimal degree, radian, or grad mode for angles. COS 60 is 0.5 in the decimal degree mode.

1 **CUR** numeric expression

CUR is a numeric function which returns cubic root of its argument. CUR 8 is 2.

1 **DEG** numeric expression

The DEG function converts an angle argument in DMS (Degrees, Minutes, Seconds) format to DEG (Decimal Degrees) form. In DMS format the integer portion of the number represents degrees, the first and second digits of the decimal represent minutes, the third and fourth digits of the decimal represent seconds, and any further digits represent decimal seconds. For example, 55° 10′ 44.5″ is represented as 55.10445. In DEG format the integer portion is degrees and the decimal portion is decimal degrees. DEG 55.10445 is 55.17902778.

1 **DMS** numeric expression

DMS is a numeric function which converts an angle argument in DEG format to DMS format (see DEG). DMS 55.17902778 is 55.10445.

1 **EXP** numeric expression

EXP is a numeric function which returns the value of e (2.718281828—the base of the natural logarithms) raised to the value of the numeric argument. EXP 1 is 2.718281828.

1 **FACT** numeric expression

FACT is a numeric function which returns the factorial of its argument. FACT 5 is 120.

1 **HCS** numeric expression

HCS is a numeric function which returns the hyperbolic cosine of the numeric argument. HCS 5 is 74.20994852.

1 **HSN** numeric expression

HSN is a numeric function which returns the hyperbolic sine of the numeric argument. HSN 4 is 27.2899172.

1 **HTN** numeric expression

HSN is a numeric function which returns the hyperbolic tangent of the numeric argument. HTN 1 is 0.761594156.

1 **INT** numeric expression

INT is a numeric function which returns the integer part of its numeric argument. INT PI is 3.

1 **LN** numeric expression

LN is a numeric function which returns the logarithm to the base e (2.718281828) of its numeric argument. LN 100 is 4.605170186).

1 **LOG** numeric expression

LOG is a numeric function which returns the logarithm to the base 10 of its numeric argument. LOG 100 is 2.

1 **POL** (numeric expression, numeric expression)

POL is a numeric function which converts numeric arguments in rectangular coordinates format to polar coordinate format.
The first numeric argument indicates the distance from the y-axis and the second numeric argument indicates the distance from the x-axis. The values converted, the distance and the angle in the polar coordinates, are assigned to the fixed variables Y and Z, respectively. The angle converted depends on whether the **COMPUTER** is in decimal degree, radian, or grad mode for angles. POL (3, 4) is (5, 53.13010235) in decimal degrees.

1 **RCP** numeric expression

RCP is a numeric function which returns the reciprocal of its numeric argument. RCP 5 is 0.2.

1 **REC** (numeric expression, numeric expression)

REC is a numeric function which converts numeric arguments in polar coordinates format to rectangular coordinates format.
The first numeric argument indicates the distance and second numeric argument indicates the angle which depends on whether the **COMPUTER** is in decimal degree, radian, or grad mode for angles. The values converted, the distances from the y-axis and the x-axis, are assigned into the fixed variables Y and Z, respectively. REC (7, 50) is (4.499513268, 5.362311102) in decimal degrees.

1 **RND** numeric expression

RND is a numeric function which generates random numbers. If the value of the argument is less than one but greater than or equal to zero, the random number is less than one and greater than or equal to zero. If the argument is an integer greater than or equal to 1, the result is a random number greater than or equal to 1 and less than or equal to the argument. If the argument is greater than or equal to 1 and not an integer, the result is a random number greater than or equal to 1 and less than or equal to the smallest integer which is larger than the argument. (In this case, the generation of the random number changes depending on the value of the decimal portion of the argument.)

	Result	
Argument	Lower Bound	Upper Bound
.5	Ø<	<1
2	1	2
2.5	1	3

The same sequence of random numbers is normally generated because the same "seed" is used each time the **COMPUTER** is turned on. To randomize the seed, see the RANDOM verb.

1 numeric expression **ROT** numeric expression

ROT is a numeric function which returns the power root of its argument.
125 ROT 3 is 5.
(i.e.: $\sqrt[3]{125}$ should be entered as 125 ROT 3.)

1 **SGN** numeric expression

SGN is a numeric function which returns a value based on the sign of the argument. If the argument is positive, the result is 1; if the argument is zero, the result is 0; if the argument is negative, the result is −1. SGN −5 is −1.

1 **SIN** numeric expression

SIN is a numeric function which returns the sine of the angle argument. The value returned depends on whether the **COMPUTER** is in decimal degree, radian, or grad mode for angles. SIN 30 is 0.5 in decimal degrees.

1 **SQR** numeric expression

SQR is a numeric function which returns the square root of its argument. It is identical to the use of the special square root symbol ($\sqrt{\ }$) on the keyboard. SQR 4 is 2.

1 **SQU** numeric expression

SQU is a numeric function which returns the square of its numeric argument. SQU 3 is 9.

1 **TAN** numeric expression

TAN is a numeric function which returns the tangent of its angle argument. The value returned depends on whether the **COMPUTER** is in decimal degree, radian, or grad mode for angles. TAN 45 is 1 in decimal degrees.

1 **TEN** numeric expression

TEN is a numeric function which returns the value of 10 (the base of the common logarithms) raised to the value of its numeric argument.
TEN 3 is 1000.

String Functions

String functions are a group of operations used for manipulating strings. Some take a string argument and return a numeric value. Some take a string argument and return a string. Some take a numeric value and return a string. Some take a string argument and one or two numeric arguments and return a string. Many dialects of BASIC require the argument of a function to be enclosed in parentheses. The **COMPUTER** does not require these parentheses, except when it is necessary to indicate what part of a more complex expression is to be included in the argument. String functions with two or three arguments all require the parentheses.

1 **ASC** string expression

ASC is a string function which returns the numeric character code value of the first character in its argument. The chart of character codes and their relationship to characters is given in Appendix B. ASC "A" is 65.
The **COMPUTER** uses ASCII codes and their characters.

1 **CHR$** numeric expression

CHR$ is a string function that returns the character which corresponds to the numeric character code of its argument. The chart of character codes and their relationship to characters is given in Appendix B. CHR$ 65 is "A".

1 **LEFT$** (string expression, numeric expression)

LEFT$ is a string function which returns the leftmost part of the string in the first argument. The number of characters returned is determined by the numeric expression. LEFT$ ("ABCDEF", 2) is "AB".

1 **LEN** string expression

LEN is a string function which returns the length of the string argument. LEN "ABCDEF" is 6.

1 **MID$** (string expression, num. exp. 1, num. exp. 2)

MID$ is a string function which returns a middle portion of the string in the first argument. The first numeric argument indicates the first character position to be included in the result. The second numeric argument indicates the number of characters that are to be included. MID$ ("ABCDEF", 2,3) is "BCD".

1 **RIGHT$** (string expression, numeric expression)

RIGHT is a string function which returns the rightmost part of the string in the first argument. The number of characters returned is determined by the numeric expression. RIGHT$ ("ABCDEF", 3) is 'DEF'.

1 **STR$** numeric expression

STR$ is a string function which returns a string which is the character representation of its numeric argument. It is the reverse of VAL. STR$ 1.59 is '1.59'.

1 **VAL** string expression

VAL is a string function which returns the numeric value of its string argument. It is the reverse of STR$. The VAL of a nonnumber is zero. VAL "1.59" is 1.59.

Note: The character string convertible by VAL function to a numerical value consists of numerals (0 to 9), symbols (+ and −) and a symbol (E) indicating an exponential portion. No other characters and symbols are included. If a character string includes other characters and symbols, any character string on the right of that character string will be ignored. If included in a character string, a space is usually regarded as nonexistent.

Serial I/O Related Commands

1 **CLOSE** #1

Abbreviations: CLOS.#1

See also: OPEN

Purpose

Closes the circuit of the serial I/O interface.

Use

This command closes the circuit (in the software sense) of the serial I/O interface which was opened by the OPEN command.
Therefore, after this command is executed, any output to the serial I/O terminal or input from the same terminal cannot be performed.

Note: This command has the same effect as the CLOSE command. In other words, you may omit #1 after CLOSE.

1 **CONSOLE** expression

Abbreviations: CONS., CONSO., CONSOL.

See also: OPEN, LPRINT, LLIST

Purpose

Sets the number of columns per line for data sending.

Use

This command sets the number of columns per line for data sent from the serial I/O interface (terminal) using the LPRINT or LLIST command.
The **COMPUTER** sends an end code (CR, LF, or CR + LF) after sending the preset line of data.

* Valid values of the expression are integers in the range of 1 to 160. If the value of the expression exceeds 160, 160 columns per line will be set. An error (ERROR 3) results if the value is 0 or negative.
  If an expression is not specified, the command is ignored and the number of columns previously set is retained.
* The number of columns becomes 39 when the batteries are replaced or when the RESET button is pressed.

1 **INPUT** #1 variable, variable, variable...

Abbreviations: I.#1, IN.#1, INP.#1, INPU.#1

See also: OPEN, PRINT#1

Purpose

Assigns data, input through the serial I/O interface (terminal), to the specified variables.

Use

* This command is valid only when the circuit of the serial I/O interface is open (after the OPEN command is executed) and is ignored otherwise.

* The INPUT#1 command assigns data (sent in the form as described in the section on the PRINT#1 command) to the specified variables.
  Therefore, the variables are specified as in the PRINT#1 command.

Example

INPUT#1A, AB, C$, E(*)

Data input through the I/O interface is assigned to variables A, AB, and C$, and array variable E( ).

* Be sure that the type of both the specified variables and the input data match (i.e., character or numeric types).
  In the ASCII code system, if a character is assigned to a numeric variable, its value becomes 0. If a number is assigned to a character variable, its contents become a character string. Therefore, if the types of the specified variable and the input data do not match, unexpected values may result.

  Even if data in a form such as function "SIN 30" is given to a numeric variable, it is assumed to be a character string with its content 0.
  For data in the form of "10+40", the characters (numbers) after the operator symbol are ignored. Therefore, the data in this case is "10".

Notes: 1. If CR (control code: ØDH) or NULL (ØØH) is included within the input data, all data following it may be ignored.

2. Simple variables and array variables must be allocated in the program/data area before executing an INPUT#1 command. An error will result if these variables are not allocated.

1 **LLIST**

2 **LLIST** {expression / "label"}

3 **LLIST** expression 1, expression 2

Abbreviations: LL., LLI., LLIS.

See also: OPEN, CONSOLE

Purpose

Sends the program contents out of the serial I/O interface (terminal).

Use

The LLIST command is valid under manual operation in the PRO or RUN mode. When the circuit of the serial I/O interface is open due to the OPEN command, the program is sent out in ASCII code.
When the circuit is closed, the program is printed on the printer. (See page 136.)

* In format 1, all programs in the **COMPUTER** are sent out.

Example:
When the program below is in the **COMPUTER**, pressing

LLIST [ENTER]

sends out the program in the form shown below.

```
 10: OPEN
100: REM  **ABC-12**
65279: END
```

Space	1	0	:	O	P	E	N	Space	CR
1	0	0	:	R	E	M	Space	Space	*
*	A	B	C	-	1	2	*	*	CR
6	5	2	7	9	:	E	N	D	Space
CR									

Note: CR is an end code. It is either LF or CR + LF, depending on the setting of the OPEN command.

* In format 2, the line indicated by the value of the expression or the line with the specified label is sent out.
* In format 3, the program, from the line indicated by the value of expression 1 to the line indicated by the value of expression 2, is sent out. (Labels can also be used for expression 1 and expression 2.)
  Expression 1 or expression 2 can be omitted in format 3.
* If expression 1 is omitted, the program, from the first line to the line indicated by the value of expression 2, is sent out.
* If expression 2 is omitted, the program, from the line indicated by the value of expression 1 to the last line, is sent out.

* If a line corresponding to the value of expression, expression 1 or expression 2 does not exist, the line with the next largest number which does exist will be specified. An error results (ERROR 1) if the lines specified in expression 1 and expression 2 are the same.

* The LLIST command is ignored if a password has been set.

* If programs have been merged using the MERGE command, the LLIST command functions only for the last merged program.
  To list the previously stored programs, execute

  LLIST "label"

* The number of print columns per line is set by the CONSOLE command. If set to 23 columns or less, executing the LLIST command results in an error (ERROR 3).

* "π" is converted to PI and "$\sqrt{\ }$" to SQR before being sent.

Note: LLINE, RLINE, GLCURSOR, LTEXT, GRAPH, LF, CSIZE, COLOR, SORGN, CROTATE, CIRCLE, or PAINT verb should be written at the beginning of a program line following the line number. If any of these verbs is written at any positions after the 24th character of the program line, the desired graphic data will not be properly output on the CE-140P printer.

1 LOAD

Abbreviation: LOA.

See also: OPEN, CLOAD

Purpose

Loads the data sent from the serial I/O interface (terminal) into the program/data area.

Use

The LOAD command is valid when the circuit of the serial I/O interface is open due to the OPEN command. It is ignored when the circuit is closed.

ASCII codes

* Data through the serial I/O interface is read until the end code is reached. Data until the end code is considered to be the first line of the program. The **COMPUTER** converts the data into a form which can be stored as a program and then transfers (writes) it to the program/data area. Then, data is again read from the serial I/O interface, converted in the same way, and then written into the memory. This operation continues until the text end code (see OPEN command) is read.
* Up to 256 bytes of data can be read at a time. Therefore, if more than 256 bytes of data are sent before the end code is read, an error results.
* The data which has been read is converted and then written to the program/data area. If one line, including the line number, exceeds 80 bytes, an error results. Further, an error also results if the beginning of the line is not a numeric value (line number).
* During execution of the LOAD command, the order of the lines is not rearranged (e.g., ascending order of the line numbers).

Note:

Execution of the LOAD command ends when the text end code is read (from the sending side).

Even if the sending side sends out the entire program, the **COMPUTER** does not end execution as long as the text end code is not read. In this case, end the execution as follows:

(1) After the sending side sends the program, have it also send only the text end code.

(2) Or, press the BRK key to end execution.

Intermediate codes

* Data is read as intermediate codes and written to the program/data area.
* When the execution of LOAD is broken or interrupted due to an error, the entire program will be erased. (Variables will be retained.)

1 **LPRINT** {expression / character string}

2 **LPRINT** {expression / character string}, {expression / character string}, ... , {expression / character string}

3 **LPRINT** {expression / character string} ; {expression / character string} ; ... ; {expression / character string}

4 **LPRINT** {expression / character string} ;

(Format where a " ; " is added to the end of 1 and 3 above.)

5 **LPRINT**

Abbreviations: LP., LPR., LPRI., LPRIN.

See also: OPEN, CONSOLE, USING

Purpose

Sends the specified information out through the serial I/O interface (terminal).

Use

When the circuit of the serial I/O interface is opened by the OPEN command, the specified information is sent out through the serial I/O terminal in ASCII code. When the circuit is closed, LPRINT prints information on the printer. (See page 169.)

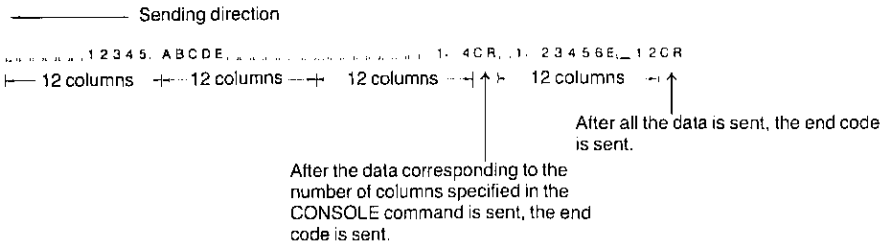
* In format 1, the value of the expression or the character string is sent from its beginning.
  If the value of the expression is negative, a "−" sign is sent before the value. If positive, a space is sent.

* In format 2, 12-column (digit) divisions are automatically set. The value of a single expression or a character string is sent within a range of 12 columns (digits).

Example:

```
10 OPEN "1200, N, 8, 1, A, C"
20 CONSOLE 36
30 LPRINT 12345, "ABCDE",
   -7/5, 1. 23456789E12
```

Executing this program sends information in the following form.


Sending direction
12345, ABCDE, 1, 4CR, 1, 23456E, 12CR
12 columns
12 columns
12 columns
12 columns
After all the data is sent, the end code is sent.
After the data corresponding to the number of columns specified in the CONSOLE command is sent, the end code is sent.

If the specified character string exceeds 12 columns in this format, only the first 12 characters are sent.
Also, if the value of the expression exceeds 12 digits (in exponential display), it is sent after the low-order digits of the fractional part are truncated.
If the value of the expression is negative, a "−" signal is sent before the value. If positive, a space is sent.

* In format 3, the specified values or strings are sent in the specified sequence. In this format, a space is not sent before a positive number.

  Example:

```
:
50 LPRINT −123; "ABC"; 567. 89
← Sending direction
−123. ABC567. 89CR
```

  End code (LF or CR + LF is sent depending on how the OPEN command is specified.)

* In format 4, the end code which indicates the end of the data is not sent. However, after the data corresponding to the number of columns specified in the CONSOLE command is sent, the end code is sent.

* In format 5, only the end code is sent.

* When the format has been specified in the USING command, formats 1 to 4 send data accordingly.

* "π" is converted to PI and "$\sqrt{\ }$" to SQR before being sent.

* Executing PRINT=LPRINT causes the PRINT command to function as LPRINT command. Specifying PRINT=LPRINT is valid only if it is executed when the printer is connected to the printer terminal or when the circuit of the serial I/O interface is open due to the OPEN command.

Note: To send characters or control codes which cannot be directly entered through the keyboard, specify them using the CHR$ command as shown below.

Example: To send [ ]

①
```
50 LPRINT CHR$&5B; CHR$&5D
```

②
```
50 A$=CHR$&5B:B$=CHR$&5D
60 LPRINT A$; B$
```

NULL (ØØH) is valid only in ① and will be ignored in ②.

Note: Do not use any BASIC command or verb as a character string in an LPRINT statement.

1 **OPEN** "baud rate, parity, word length, stop bit, type of code, end code, text end code"

2 **OPEN**

Abbreviations: OP., OPE.

See also: CLOSE

Purpose

Allows data to be transferred through the I/O interface. Also sets the I/O conditions.

Use

Format 1 enables data to be transferred through the I/O interface (serial I/O terminal). It also sets the conditions for the data transfer with the connected equipment. The conditions are specified in the following form:
"baud rate, parity, word length, stop bit, type of code, end code, text end code"

Baud Rate: 300, 600, 1200
Specifies the modulation rate (transfer rate). For the **COMPUTER**, 300 baud, 600 baud, or 1200 baud (bps) can be selected.
(1 baud = 1 bit/sec)

Parity: N, E, O
Specifies the type of parity by a character.
N: No parity bit is transmitted nor received.
E: Specifies even parity.
O: Specifies odd parity.

Word Length: 7, 8
Specifies how many bits to be transmitted or received per character. Either 7 or 8 bits can be specified.

Number of Stop Bits: 1, 2

Type of Code: A, B
Specifies code system to be transmitted and received.
A: Specifies ASCII codes.
B: Specifies intermediate codes.
Word length cannot be set to 7 bits with intermediate codes; results in ERROR 1.

End Code: C, F, L
Specifies the type of end code to indicate the end of data (delimiting), end of a program line, etc.
C: Specifies the CR (carriage return) code.
F: Specifies the LF (line feed) code.
L: Specifies the CR code + LF code.

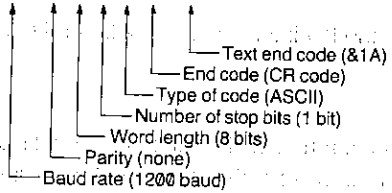
Text End Code: &ØØ-&FF

Specifies the text end code to indicate the end of the program, etc.

(May be required when using the SAVE or LOAD commands.)

Example

```
OPEN "1200,N,8,1,A,C,&1A"
```


Text end code (&1A)
End code (CR code)
Type of code (ASCII)
Number of stop bits (1 bit)
Word length (8 bits)
Parity (none)
Baud rate (1200 baud)

The conditions in the example above are set after the batteries are replaced or after the RESET button is pressed.

* Any condition specified in the OPEN command can be omitted. If omitted, the current condition remains unchanged.

Example

```
OPEN ",,,2"
```

Only the number of stop bits is changed.

* In format 2, all conditions set previously are retained. This format enables data to be transferred through the I/O interface.
* Executing the OPEN command while the circuit of the I/O interface is open and ready for data transfer (due to prior execution of the OPEN command) results in an error (ERROR 8).
  Execute the CLOSE command to close the circuit. (The circuit also closes when the RUN command is executed, when the program ends, when the power is switched on, or when the computer is in the CAL mode.)
  The previously set conditions are retained even after the CLOSE command is executed.

1 **OPEN$**

Abbreviations: OP.$, OPE.$

See also: OPEN

Purpose

Obtains the currently set I/O conditions.

Use

The currently set I/O conditions are obtained as a character string.

Example

OPEN$ [ENTER] 1200, N,8, 1, A, C, &1A

Note: This command is used to verify the serial I/O conditions, and therefore, cannot be used in combination with any other function.
When using the command in a program, write as PRINT OPEN$.

1 **PRINT#1** variable, variable, variable ...

Abbreviations: P.#1, PR.#1, PRI.#1, PRIN.#1

See also: OPEN, INPUT#1, PRINT#

Purpose

Sends the contents of the specified variables through the serial I/O interface (terminal).

Use

This command is valid only when the circuit of the serial I/O interface is open (due to the OPEN command). It is ignored otherwise.

ASCII codes

* Variables are specified as follows.

Fixed Variables:	Specify each variable name. Example: A, B, C$ Note: Fixed variables cannot be specified in the form of A*.
Simple Variables:	Specify each variable name. Example: AA, B1$, C2
Array Variables:	Specify in the form of array name (*). Example: B(*), C$(*) Specified in this manner, the contents of all elements in the array are sent. (Array elements cannot be specified individually.) Example: 50 PRINT #1A, AB, C$, E(*)

* When data is sent, the end code is added to the end of the contents of each variable. The end code is added to the end of the contents of each element for array variables also.

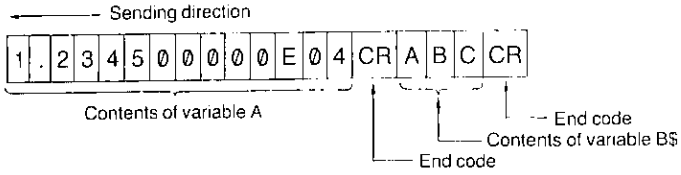
Example:

When A = 12345 and B$ = "ABC",

executing

```
PRINT#1A,B$
```

sends A and B$ in the following form (provided the end code is CR).



* If the value of the numeric variable is negative, a "−" sign is sent before the value.

* The elements of an array are sent in the following sequence.

  1 dimensional array

  Example: For B (3)

  B (Ø) → B (1) → B (2) → B (3)

  2 dimensional array

  Example: For C (2, 3)

  C (Ø, Ø) → C (Ø, 1) → C (Ø, 2) .....

Intermediate codes

* Variable specification is virtually the same as in ASCII except that fixed variables can also be specified as A*. (Refer to PRINT# command.)
* Variables are sent according to the internal format, but the end code is not sent. Also, a label (total number of bytes sent, string elements, others) will be attached to the beginning of each variable. The CR code will be sent at the end when variables are specified in the A* format.

Notes: 1. The locations for extended variables such as A(27) and beyond, simple variables, and/or array variables must be allocated in the program/data area before the PRINT#1 command is executed. An error results if an attempt is made to send the contents of a variable which has not been allocated.
An error also results if the type of the specified variable (numeric or character) and that of within the **COMPUTER** do not match.

2. If symbols such as "π" and "$\sqrt{\ }$" are included in the data to be sent, they are converted to "PI" and "SQR," respectively, and then sent.

1 **SAVE**

Abbreviations: SA., SAV.

See also: OPEN, LLIST

Purpose

Sends the program within the **COMPUTER** out through the serial I/O interface (terminal).

Use

When the circuit of the serial I/O interface is open due to the OPEN command, the program is sent.
The command is ignored if the circuit is closed.

* The SAVE command is ignored if a password has been set.
* The following differences exist between the two code systems.

ASCII: Sends an end code at the end of each line.
Sends a text end code at the end of text transmission.

Intermediate: Does not send an end code or a text end code.

TEXT Functions

1 **BASIC**

Abbreviations: BA., BAS., BASI.

See also: TEXT

Purpose

Clears the text mode.
(valid only in manual operation in the PROgram mode)

Use

Executing this command clears the Text mode and returns the mode to BASIC. As the mode returns to BASIC, the prompt symbol changes from "<" to ">".

* Changing from the Text mode to the BASIC mode usually changes the text in the **COMPUTER** to a program (internal code).

  All lowercase alphabetics other than those in character strings enclosed in quotation marks (" ") or following a REM statement are automatically converted to uppercase alphabetics.

  However, abbreviations such as "P." and "I." are not converted to their respective commands. (In this case, call the cursor to the line and press the [ENTER] key to convert it to a command.)

* Due to characteristics of the text function, commands and formats included in the text but not found in the computer may not be executable.

* During program conversion, "**" are displayed at the right end of the display unit.

* If a password has been set, executing the BASIC command results in an error (ERROR 1).

1 **TEXT**

Abbreviations: TE., TEX.

See also: BASIC

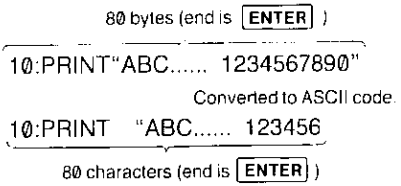
Purpose

Sets the Text mode.
(valid only in manual operation in the PROgram mode)

Use

The text function is used when inputting a program written for a higher-level personal computer. The program input by the **COMPUTER** is sent to the host through the serial I/O interface.

* Executing the TEXT command sets the Text mode. In the Text mode, a number corresponding to the line number and then information corresponding to program commands or data are entered. Then the [ENTER] key is pressed to write the input to the program/data area.
  However, the written contents, unlike in the BASIC mode, are not converted to commands (internal codes). The text is stored as it is (as characters and/or numbers) in ASCII codes. The text is arranged in the order of the numbers corresponding to the line number at the beginning of each line. (Line number editing function.)
* The text written in the Text mode is stored as it is. Therefore, command abbreviations in BASIC (such as I. for INPUT) are displayed and stored as they are.
* If a program is stored in the internal code of the **COMPUTER** with the text mode set, it is converted to ASCII code.
* During program conversion, "**" are displayed at the right end of the display unit.
* The prompt symbol is "<" in the Text mode. (It is usually ">".)
* The line in the Text mode (including line number and [ENTER]) must not exceed 80 characters (80 bytes). If a line exceeds 80 characters due to program conversion, the excess part will be cleared.



In this example, the PRINT command is one byte in internal code but takes up 6 bytes in ASCII code. Because of this, the last few characters (7890) are deleted.

* The number of bytes increases when converting a program from internal code to ASCII code, as shown in the example. If, as a result, the capacity of the program area is exceeded, the program converted up to that point is converted back to internal code and an error will result (ERROR 6).

* If a password has been set, an error (ERROR 1) occurs when the TEXT command is executed.

CHAPTER 10
TROUBLESHOOTING

This chapter provides you with some hints on what to do when your **SHARP COMPUTER** does not do what you expect from it. It is divided into two parts—the first part deals with general machine operation and the second with BASIC programming. For each problem, there are a series of remedies suggested. You should try each of these, one at a time, until you have solved the problem.

Machine Operation

If:	Then You Should:
There is nothing on the display after you have turned on the machine	1. Check to see that the power switch is in the ON position. 2. Press the ON BRK key to see if AUTO POWER OFF has been activated. 3. Replace the batteries. 4. Adjust the contrast control. 5. Check whether the LOCK switch for the RAM card compartment is in locked position.
There is a display, but no response to keystrokes	1. Press C·CE key to clear. 2. Press CA (SHIFT C·CE) key to clear. 3. Turn the power OFF and ON again. 4. Hold down any key and push RESET button. 5. Push RESET button without pressing any key.
You get no response after you have typed in a calculation or answer	1. Press ENTER key.
The machine displays something and stops while you are running a BASIC program	1. Press ENTER key.
Your calculation entered is displayed in BASIC statement format (colon after the first number)	1. Change the mode from PROgram to RUN for calculations.

If:	Then You Should:
You get no response from any keys	1. Hold down any key and push RESET button. 2. If you get no response from any key even after the above operation, push RESET button. Then, press [Y], [=], or [ENTER] in response to "MEMORY CLEAR O.K.?" message. This will clear the programs and data in memory.

BASIC Debugging

When entering a new BASIC program, it is usual for it not to work the first time. Even if you are simply keying in a program that you know is correct, such as those provided in this manual, it is usual to make at least one typing error. If it is a new program of any length, it will probably contain at least one logic error as well. Following are some general hints on how to find and correct your errors.

You run your program and get an error message:

1. Go back to the PROgram mode and use the ↑ or the ↓ key to recall the line with the error. The cursor will be positioned at the place in the line where the **COMPUTER** got confused.
2. If you can't find an obvious error in the way in which the line is written, the problem may lie with the values which are being used. For example, CHR$(A) will produce a space if A has a value of 1. Check the value of each variable in either the RUN or the PROgram mode by typing in the name of the variable followed by ENTER .

You RUN the program and don't get an error message, but it doesn't do what you expect.

3. Check through the program line by line using LIST and the ↓ and ↑ keys to see if you have entered the program correctly. It is surprising how many errors can be fixed by just taking another look at the program.
4. Think about each line as you go through the program as if you were the computer. Take sample values and try to apply the operation in each line to see if you get the result that you expected.
5. Insert one or more extra PRINT statements in your program to display key values and key locations. Use these to isolate the parts of the program that are working correctly and the location of the error. This approach is also useful for determining which parts of a program have been executed. You can also use STOP to temporarily halt execution at critical points so that several variables can be examined.
6. Use TRON and TROFF, either as commands or directly within the program to trace the flow of the program through individual lines. Stop to examine the contents of critical variables at crucial points. This is a very slow way to find a problem, but sometimes it is also the only way.

CHAPTER 11
MAINTENANCE OF THE COMPUTER

To insure trouble-free operation of your **SHARP COMPUTER**, we recommend the following:

* Always handle the **COMPUTER** carefully as the liquid crystal display is made of glass.
* Keep the computer in an area free from extreme temperature changes, moisture, or dust. During warm weather, vehicles left in direct sunlight are subject to high temperature buildup. Prolonged exposure to high temperature may cause damage to your computer.
* Use only a soft, dry cloth to clean the computer. Do not use solvents, water, or a wet cloth.
* To avoid battery leakage, remove the batteries when the computer will not be used for an extended period of time.
* If service should be required on this equipment, use only a SHARP servicing dealer, SHARP approved service facility, or SHARP repair service where available.
* If the computer is subjected to strong static electricity or external noise, it may "hang up" (all keys become inoperative). If this occurs, press the RESET button while holding down any key. (See Troubleshooting.)
* Keep this manual for further reference.

APPENDIX A
Error Messages

There are nine different error codes built into the **COMPUTER**. The following table will explain these codes.

Error Number **Meaning**

1 Syntax error

- This means that the **COMPUTER** can't understand what you have entered. Check for things such as semicolons on the ends of PRINT statements, misspelled words, and incorrect usages.

```
3*/2
```

2 Calculation error

Here you have probably done one of three things:

1. Tried to use too large a number.
   Calculation results are greater than 9.999999999E 99.
2. Tried to divide by zero.

```
5/0
```

3. An illogical calculation has been attempted.

```
LN -30 or ASN 1.5
```

3 DIMension error/Argument error

- Array variable already exists.

  Array specified without first dimensioning it.

  Array subscript exceeds size of array specified in DIM statement.

```
DIM B(256)
```

- Illegal function argument. This means that you have tried to make the **COMPUTER** do something that it just can't handle.

  The time interval specified for the WAIT verb is greater than 65535.

```
WAIT 66000
```

  When the LLIST command is used for serial I/O interface output, the number of print columns per line set by the CONSOLE command is 23 columns or less.

4 Line number error

Here you have probably done one of two things:

1. Tried to reference an nonexistent line number with a GOTO, GOSUB, RUN, LIST, THEN, or the like.
2. Tried to use too large a line number. The maximum line number is 65279.

5 Nesting error

Subroutine nesting exceeds 1Ø levels.

FOR loop nesting exceeds 5 levels.

RETURN verb without a GOSUB, NEXT verb without a FOR, or READ verb without a DATA.

Buffer space exceeded.

6 Memory overflow

Generally this error happens when you've tried to DIMension an array that is too big for memory. This can also happen when a program becomes too large.

7 PRINT USING error

This means that you have put an illegal format specifier into a USING statement.

8 I/O device error

This error can happen only when you have the optional printer and/or cassette recorder connected to the **COMPUTER**. It means that there is a problem with communication between the I/O device and the **COMPUTER**.

9 Other errors

This code will be displayed whenever the computer has a problem that isn't covered by one of the other eight error codes. One of the most common causes for this error is trying to access data in a variable in one fashion (e.g., A$) while the data was originally stored in the variable in another fashion (e.g., A).

ERRORS RELATED TO RENUM

Error message	Description
ERROR 1	A syntax error exists in the RENUM command.
ERROR 1 IN line number	A line number reference is missing from the command specifying the jump destination (e.g., GOTO, GOSUB, etc.).
ERROR 3 IN line number	A line number greater than 65279 is encountered during the execution of the RENUM command. The length of one program line exceeds 79 bytes.
ERROR 4	The specified old line number does not exist in the program.
ERROR 4 IN line number	The line number specified as the jump destination does not exist in the program file.
ERROR 6	Memory capacity is insufficient to execute the RENUM command, or becomes short during the renumbering process.
ERROR 9	An attempt was made to execute the RENUM in other than PRO (Program) mode. An attempt was made to change the execution order of program lines by specifying the new line number lower than the line number immediately before the old line number.
ERROR 9 IN line number	The line number specified as the jump destination is inappropriate, because it uses a variable, expression, or function (i.e., incorrect line number reference).

APPENDIX B
CHARACTER CODE CHART

The following chart shows the conversion values for use with CHR$ and ASC. The column shows the first hex character or the first four binary digits (i.e., bits); the row shows the second hex character or the second four bits. The upper left corner of each box contains the decimal number for the character. The lower right shows the character. If no character is shown, then it is an illegal character on the **COMPUTER**.

For example, the character "A" is decimal 65 or hex 41 or binary 01000001. The character '√' is decimal 252 or hex FC or binary 11111100.

Notes:

- The characters for character codes 92(&5C), 249(&F9), and 250(&FA) appearing on the computer display differ from the characters for these codes printed by optional printers CE-126P and CE-140P.
- The character printed for character code 92(&5C) shown in the following table is ◣ with the CE-126P printer, and \ (back slash) with the CE-140P printer.
- When using the CE-126P printer, do not use code 0(&00).
- For printing characters on the CE-140P, codes 8 (&08), 10 (&0A), 11 (&08), 13 (&0D), and 27 (&1B) are the control codes for the printer.
  The rest, 0 (&00) through 31 (&1F), are NULL.
- Any codes other than 0 (&00) through 31 (&1F) not used for characters are printed as spaces.
- Codes 249 (&F9) and 250 (&FA) are spaces.
- For printing characters on the CE-515P, refer to the Character Code Table in the CE-515P Operation Manual and specify the character code.

APPENDIX B
Character Code Chart

First 4 Bits

Second 4 Bits: Hex / Binary	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	E 1110	F 1111
0 0000	0 NUL	16	32 SPACE	48 0	64 @	80 P	96 `	112 p	128	224	240
1 0001	1	17	33 !	49 1	65 A	81 Q	97 a	113 q	129	225	241
2 0010	2	18	34 "	50 2	66 B	82 R	98 b	114 r	130	226	242
3 0011	3	19	35 #	51 3	67 C	83 S	99 c	115 s	131	227	243
4 0100	4	20	36 $	52 4	68 D	84 T	100 d	116 t	132	228	244
5 0101	5	21	37 %	53 5	69 E	85 U	101 e	117 u	133	229	245 ♠
6 0110	6	22	38 &	54 6	70 F	86 V	102 f	118 v	134	230	246 ♥
7 0111	7	23	39 '	55 7	71 G	87 W	103 g	119 w	135	231	247 ♦
8 1000	8	24	40 (	56 8	72 H	88 X	104 h	120 x	136	232	248 ♣
9 1001	9	25	41)	57 9	73 I	89 Y	105 i	121 y	137	233	249 ▉
A 1010	10	26	42 *	58	74 J	90 Z	106 j	122 z	138	234	250 ⌐
B 1011	11	27	43 +	59 ;	75 K	91 [	107 k	123 {	139	235	251 π
C 1100	12	28	44 ,	60 <	76 L	92 \	108 l	124 ¦	140	236	252 √
D 1101	13	29	45 –	61 =	77 M	93]	109 m	125 }	141	237	253
E 1110	14	30	46 .	62 >	78 N	94 ^	110 n	126 ~	142	238	254
F 1111	15	31	47 /	63 ?	79 O	95 _	111 o	127	143	239	255

APPENDIX C
FORMATTING OUTPUT

It is sometimes important or useful to control the format as well as the content of the output. The **COMPUTER** controls display formats with the USING verb. This verb allows you to specify:

- the number of digits
- the location of the decimal point
- the scientific notation format
- the number of string characters

These different formats are specified with an "output mask". This mask may be a string constant or a string variable:

```
1Ø: USING "####"
2Ø: M$="&&&&&&"
3Ø: USING M$
```

When the USING verb is used with no mask, all special formatting is canceled.

```
4Ø: USING
```

A USING verb may also be used within a PRINT statement:

```
5Ø: PRINT USING M$; N
```

Wherever a USING verb is used, it will control the format of all output until a new USING verb is encountered.

Numeric Masks

A numeric USING mask may only be used to display numeric values, i.e., numeric constants or numeric variables. If a string constant or variable is displayed while a numeric USING mask is in effect, the mask will be ignored. A value that is to be displayed must always fit within the space provided by the mask. The mask must reserve space for the sign character, even when the number will always be positive. Thus a mask that shows four display positions may only be used to display numbers with three digits.

Specifying Number of Digits

The desired number of digits is specified using the '#' character. Each '#' in the mask reserves space for one digit. The display or print always contains as many characters as are designated in the mask. The number appears to the far right of this field; the remaining positions to the left are filled with spaces. Positive numbers therefore always have at least one space at the left of the field. Since the **COMPUTER** maintains a maximum of 10 significant digits, no more than 11 '#' characters should be used in a numeric mask.
When the total number of columns of the integer part specified exceeds 11, this integer part is regarded as 11 digits in the **COMPUTER**.

NOTE: In all examples in this appendix, the beginning and end of the displayed field will be marked with an 'I' character to show the size of the field.

Statement	Display
10: USING "####"	(Set the **COMPUTER** to the RUN mode, type RUN, and press [ENTER].)
20: PRINT 25	| 25|
30: PRINT –350	|–350|
40: PRINT 1000	ERROR 7 IN 40

Notice that the last statement produced an error because 5 positions (4 digits and a sign space) were required, but only 4 were provided in the mask.

Specifying a Decimal Point

A decimal point character, '.', may be included in a numeric mask to indicate the desired location of the decimal point. If the mask provides more significant decimal digits than are required for the value to be displayed, the remaining positions to the right will be filled with zeros. If there are more significant decimal digits in the value than in the mask, the extra digits will be truncated (**not** rounded):

Statement	Display
10: USING "####.##"	
20: PRINT 25	| 25.00|
30: PRINT –350.5	|–350.50|
40: PRINT 2.547	| 2.54|

Specifying Scientific Notation

A "^" character may be included in the mask to indicate that the number is to be displayed in scientific notation. The '#' and '.' characters are used in the mask to specify the format of the "characteristic" portion of the number, i.e., the part which is displayed to the left of the **E**. Two '#' characters should always be used to the left of the decimal point to provide for the sign character and one integer digit. The decimal point may be included, but is not required. Up to 9 '#' characters may appear to the right of the decimal point. Following the characteristic portion, the exponentiation character, **E**, will be displayed followed by one position for the sign and two positions for the exponent. Thus, the smallest scientific notation field would be provided by a mask of "##^" which would print numbers of the form '**2E** 99'. The largest scientific notation field would be "##.#########^" which would print numbers such as '−1.234567890 **E** −12':

Statement	Display
10: USING "###.##^"	
20: PRINT 2	| 2.00E 00|
30: PRINT −365.278	|−3.65E 02|

Specifying Alphanumeric Masks

String constants and variables are displayed using the '&' character. Each '&' indicates one character in the field to be displayed. The string will be positioned at the left end of this field. If the string is shorter than the field, the remaining spaces to the right will be filled with spaces. If the string is longer than the field, the string will be truncated to the length of the field:

Statement	Display
10: USING "&&&&&&"	
20: PRINT "ABC"	|A B C |
30: PRINT "ABCDEFGHI"	|ABCDEF|

Mixed Masks

In most applications, a USING mask will contain either all numeric or all string formatting characters. However, both types of characters may be included in one USING mask for certain purposes. In such cases, each switch from numeric to string formatting characters or vice versa marks the boundary for a different value. Thus, a mask of "#####&&&&" is a specification for displaying two separate values–a numeric value which is allocated 5 positions and a string value which is allocated 4 positions:

Statement	Display
10: PRINT USING "###.##&&"; 25; "CR"	25.00CR
20: PRINT –5.789; "DB"	–5.78DB

Remember that a USING format once specified is used for all outputs which follow until canceled or changed by another USING verb.

APPENDIX D
EXPRESSION EVALUATION AND OPERATOR PRIORITY

When the **SHARP COMPUTER** is given a complex expression, it evaluates the parts of the expression in a sequence determined by the priority of the individual parts of the expression. If you enter the expression:

```
100 / 5+45
```

as either a calculation or as a part of a program, the **COMPUTER** does not know whether you meant:

$$\frac{100}{5+45} = 2 \qquad \text{or} \qquad \frac{100}{5} + 45 = 65$$

Since the **COMPUTER** must have some way to decide between these options, it uses its rules of operator priority. Because division has a higher "priority" than addition (see below), it will choose to do the division first and then the addition, i.e., it will choose the second option and return a value of 65 for the expression.

Operator Priority

Operators on BASIC mode of the **SHARP COMPUTER** are evaluated with the following priorities from highest to lowest:

Level	Operation
1	Parentheses
2	Variables and Pseudovariables
3	Functions
4	Exponentiation (^), (ROT)
5	Unary minus, negative sign (−)
6	Multiplication and division (*, /)
7	Addition and subtraction (+, −)
8	Relational operators (<, <=, =, <>, >=, >)
9	Logical operators (AND, OR, NOT, XOR)

When there are two or more operators at the same priority level, the expression will be evaluated from left to right. (The exponentiation will be evaluated from right to left.) Note that with A+B−C, for example, the answer is the same whether the addition or the subtraction is done first.

When an expression contains multiple nested parentheses, the innermost set is evaluated first and evaluation then proceeds outward.

For levels 3 and 4, the last entry has a higher priority.

For example: $-2 \wedge 4 \rightarrow -(2^4)$
$3 \wedge -2 \rightarrow 3^{-2}$

Sample Evaluation

Starting with the expression:

((3+5−2)*6+2)/10 ^ LOG 100

The **COMPUTER** would first evaluate the innermost set of parentheses. Since '+' and '−' are at the same level, it would move from left to right and would do the addition first:

((8-2)*6+2)/10 ^ LOG 100

Then it would do subtraction:

((6)*6+2)/10 ^ LOG 100

or:

(6*6+2)/10 ^ LOG 100

In the next set of parentheses, it would do the multiplication first:

(36+2)/10 ^ LOG 100

And then the addition:

(38)/10 ^ LOG 100

or:

38/10 ^ LOG 100

Now that the parentheses are cleared, the LOG function has the highest priority so it is done next:

38/10 ^ 2

The exponentiation is done next:

38/100

And last of all, the division is performed:

0.38

This is the value of the expression.

APPENDIX E
KEY FUNCTIONS IN BASIC MODE

[ON/BRK] (ON)
Used to turn the **COMPUTER** power on when the auto power off function is in effect.
(BREAK)
- Depression of this key during program execution functions as a BREAK ([ON/BRK]) key and causes to interrupt the program execution.
- When pushed during manual execution of an input/output command such as BEEP, CLOAD, etc., execution of the command is interrupted.

[SHIFT]
- This yellow key is used to designate a second function (that inscribed in brown above each key).

  Ex. [SHIFT] [?/U] → ? is input.

[C·CE]
- Used to clear the contents of the entry and the display. (Error release)

[SHIFT] [CA]
- Used to not only clear the display contents, but to reset the computer to its initial state.

  – Initial state –
  - Resets the WAIT timer.
  - Resets the display format. (USING format)
  - Resets the TRON state (TROFF).
  - Resets the PRINT = LPRINT.
  - Resets error.

[0] ~ [9]
- Numeric keys

[.]
- Decimal point
- Used to enter an abbreviation for a command, verb, or function.
- Used to designate the decimal portion in USING format designation.

[E]
- Used to designate an exponent in scientific notation. (This is the letter E key.)

[EXP]
- Used to designate an exponent in scientific notation.

[/]
- Division key

[*]
- Multiplication key
- Used to designate an array variable in the INPUT#, the PRINT#, etc.

[+] • Addition key

[−] • Subtraction key

[SHIFT] [?] • Used to enter CLOAD?

[SHIFT] [:] • Used to divide two or more statements in one line.

[,] • Used to provide pause between two equations, and between variables or comments.

[SHIFT] [;] • Used to provide multi-display (two or more values/contents/displayed at a time).
• Used to provide pause between the instruction and the variable.

[=] • Used in assignment statements to assign the contents (number or character) on the right for the variable specified on the left.
• Used when inputting logical operators in IF statement.
• When any one of eighteen keys (A, S, D, F, G, H, J, K, L, Z, X, C, V, B, N, M, ', SPaCe) is pushed after the depression of the [DEF] key, the computer starts to execute the program from the program line that has the same label as the key code depressed.

[A] ~ [Z] • Letter keys. You are probably familiar with these keys from the standard typewriter keyboard. On the **COMPUTER** display, the characters appear in the uppercase.

[(], [)] • Used to input parentheses.

[SHIFT] [<] } • Used when inputting logical operators in IF statement.
[SHIFT] [>] }

[SPC] • Used to provide space when inputting programs or characters.

[SHIFT] [^] • Used for power calculation instructions.
• Used to specify the floating decimal point system (exponent display) for numerical data in USING statement instructions.

[SHIFT] [π] • Used to designate Pi (π).

[√] • Used to designate square root.

[SHIFT] [!] • Used to designate these symbols.

["] " : • Used to designate and cancel characters.

[#] • Used to specify labels.

[$] # : Used with USING statement, to provide the instruction to define the display format of numerical data.

[%]

[&] $: • Used when assigning character variables.

[SHIFT] [@] & : • Used with USING statement, to provide the instruction to define the display format of character string.

• Used to designate hexadecimal number.

[►] • Used to shift the cursor to the right (press once to advance one position, hold down for automatic advance).
• Used to execute playback instructions.
• Used to clear an error condition in manual operation.

[◄] • Used to shift the cursor to the left (press once to advance one position, hold down for automatic advance).
• Used to execute playback instructions.
• Used to clear an error condition in manual operation.

[SHIFT] [INS] • Used to insert a space (_ appears) of 1-step capacity between the address (N) indicated by the cursor and the preceding address (N−1).

[SHIFT] [DEL] • Used to delete the contents of the address indicated by the cursor.

[SHIFT] [INPUT] • Used to preset command and verb keys. Pressing [SHIFT] and then the letter (including comma and space) key below the command or verb desired followed by [ENTER] key causes the designated command or verb to be entered into the **COMPUTER**.

[SHIFT] [CLOAD]

[CAL] • Used to set the CAL mode.

[BASIC] • Used to set the RUN mode when the CAL mode is set.
• Used to set the PRO mode when the RUN mode is set.
• The RUN and PRO modes are selected alternately each time you press the [BASIC] key.

[SHIFT] [P↔NP] • Used to set the print or nonprint mode when an optional printer is connected with the **COMPUTER**.

[SHIFT] [DRG] • Used to designate an angular unit (DEG, RAD, or GRAD).

[hyp] • Used to enter a hyperbolic function.

[SHIFT] [arc hyp] • Used to enter an inverse hyperbolic function.

[sin] ~ [x^2]
[SHIFT] [$\sin^{-1}$]
[SHIFT] [$n!$] • Used to enter a function defined in each key.

[ENTER]

- Used to enter a program line into the computer.
- Used for writing a program.
- Used to request manual calculation or direct execution of a command statement by the computer.
- Used to enter a restart instruction after inputting data required by an INPUT statement or after executing a PRINT statement.

Refer to page 47 for the keys used for MATRIX operations.

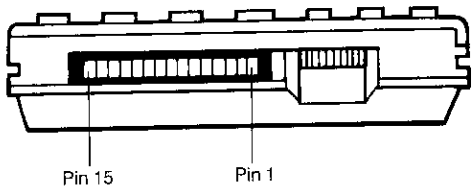
The [↑] and [↓] keys have the following functions, depending on the designated mode, as well as the state of the computer.

Mode	State	[↓]	[↑]
RUN	Program being executed		
	Program is temporarily interrupted	To execute the next line	To display the program line being executed or already executed, hold this key down.
	INPUT statement being executed		
	PRINT statement just now executed		
	Under break		
	Error condition during executing program		To display the error-producing line, hold this key down.
	TRON condition	To execute debugging operation	To display the program line being executed or already executed, hold this key down.
	Other conditions	To display an answer just previously calculated. (Last answer function)	Same as left
PRO	(When the mode is changed from RUN to PRO and program line is not being displayed)		
	Program is temporarily interrupted	To display the line interrupted	Same as left
	Error condition	To display the line with error	Same as left
	Other conditions	To display the first line	To display the last line
	(When the program line is being displayed)		
		To display the next program line	To display the preceding program line

Note: The following keys cannot be used in the BASIC mode (RUN or PRO mode). [FSE], [SHIFT] [TAB], [↕], [SHIFT] [STAT], [SHIFT] [Δ%], [x→M], [RM], [M+], [+/−], and keys used to obtain the statistics (i.e., n, $\bar{x}$, etc.)

APPENDIX F
SIGNAL USED IN THE SERIAL I/O TERMINAL

The **COMPUTER** is equipped with a 15-pin connector for the serial I/O terminals. The pins used and their signals are described below



Right side view

Pin Connections Used

Pin	Name	Symbol	I/O	Function
1	Frame Ground	FG	—	Protective chassis ground
2	Send Data	SD	Out	DC output signal
3	Receive Data	RD	In	DC input signal
4	Request to Send	RS	Out	HIGH: Sends carrier
5	Clear to Send	CS	In	HIGH: Transmission enabled
7	Signal Ground	SG	—	Reference 0 voltage for all signals
8	Data Carrier Detect	CD	In	HIGH: Carrier signal received
10		VC		Power supply
11	Receive Ready	RR	Out	HIGH: Receive enabled
12	Peripheral Acknowledge	PAK	In	See Note 3.
13		VC	—	Power supply
14	Data Terminal Ready	ER	Out	HIGH: Local terminal ready
15	Peripheral Request	PRQ	Out	See Note 3.

Notes:

1. HIGH: VC voltage level; LOW: SG voltage level.
2. The **COMPUTER** uses CMOS components. Application of voltages exceeding the allowable range, i.e., voltage level between SG and VC, may damage the computer.
3. This signal confirms if the connected terminal is the CE-140P printer or not. This signal is not used for communication with other terminals.

APPENDIX G
SPECIFICATIONS

Model: PC-1460 Pocket Computer

Processor: 8-bit CMOS CPU

Programming Language: BASIC

System ROM: 72K Bytes

Memory Capacity: RAM:

System	Approx. 1.1K Bytes
User	
Fixed Memory Area (A ~ Z, A$ ~ Z$)	208 Bytes
Program/Data Area	6878 Bytes (When using CE-212M)

Stack:

Subroutine:	10 stacks	Function:	16 stacks
FOR–NEXT:	5 stacks	Data:	8 stacks

Operators: Addition, subtraction, multiplication, division, trigonometric and inverse trigonometric functions, logarithmic and exponential functions, angle conversion, square and square root, sign, absolute, integer, relational operators, logical operators, matrix operations.

Numeric Precision: 10 digits (mantissa) + 2 digits (exponent).

Editing Features: Cursor left and right, line up and down, character insert, character delete.

Memory Protection: CMOS Battery backup.

Serial Input/Output Features:

Standards:	Start-stop transmission (asynchronous) system. Only half duplex.
Baud Rates:	300, 600, 1200 Baud (BPS)
Data Bits:	7 or 8 Bits
Parity Bits:	Even, odd, or no-parity
Stop Bit:	1 or 2 Bits
Connectors Used:	15-pin connector (for external equipment)
Output Signal Level:	CMOS level (4 to 6 Volts)
Interfacing Signals:	Inputs:RD, CS, CD, PAK Outputs: SD, RS, RR, ER, PRQ Others: SG, FG, VC

Display: 24-character Liquid Crystal Display with 5 × 7 dot pattern.

Keys: 78 keys: Alphabetics, numerics, special symbols, functions, and user defined keys.

Power Supply: 6.0V DC: Lithium cells.
Type: CR-2032×2

Power Consumption: 6.0V DC @ 0.03W
Approximately 120 hours of continuous operation under normal conditions (based on 10 minutes of operation or program execution and 50 minutes of display per hour at a temperature of 20°C). The time may vary slightly depending on usage and the type of battery used.

Operating Temperature: 0°C to 40°C

Dimensions: 182(W)×72(D)×16(H) mm.
7-5/32" (W)×2-27/32" (D)×5/8" (H)

Weight: Approximately 175 g (0.39 lb) (with two cells and one RAM card)

Accessories: Hard cover, two lithium cells (built-in), keyboard template, operation manual, and RAM Card (CE-212M).

Options: Plug-in card type 8K-Byte RAM (CE-201M),
16K-Byte RAM (CE-202M)
32K-Byte RAM (CE-203M)
2K-Byte RAM (CE-210M)
4K-Byte RAM (CE-211M)
8K-Byte RAM (CE-212M)
16K-Byte RAM (CE-2H16M)
Cassette Tape Recorder (CE-152)
Printer/Cassette Interface (CE-126P)
Printers (CE-140P, CE-515P)
Others

APPENDIX H
USING PROGRAMS WRITTEN ON OTHER PC MODELS

Programs written on the following computers can be used on the PC-1460 with slight modifications.

PC-1210 Series: PC-1210/11
PC-1245 Series: PC-1245/46/47
PC-1250 Series: PC-1250/51
PC-1260 Series: PC-1260/61
PC-1350 Series: PC-1350
PC-1401 Series: PC-1401/02
PC-1403 Series: PC-1403
PC-1450 Series: PC-1450
PC-2500 Series: PC-2500

Although the functions of these models all differ slightly, programs composed on any of these models can be used on the PC-1460 by making the modifications below.

Notes: 1. PC-1460 can read programs from tapes recorded with programs written on PC-1210 Series, PC-1245 Series, and PC-1250 Series computers, but programs written and recorded with PC-1460 cannot be read or used by computers in these three series.

2. Program tapes for the PC-1245 Series and PC-1210 Series recorded with a number of programs loaded with the MERGE command cannot be used on the PC-1460. To use them, MERGE the programs individually into the PC-1460.

3. Programs containing POKE or CALL commands written on the PC-1250 Series cannot be executed on the PC-1460. Execution of such programs may render all PC-1460 keys inoperable.

Modifications Required for PC-1245 Series (including PC-1250 Series) Programs

When using on the PC-1460 a program developed for the PC-1245 Series, the following modificiations are necessary:

(1) Multiplication without using the operator "*". On the PC-1245 Series, the operator (*) for multiplication may be omitted, such as AB for A*B or CD for C*D. On the PC-1460, however, the multiplication operator (*) cannot be omitted since the computer treats two consecutive characters, such as AB or CD, as simple variables. Use the specification on the right side of the following example:

e.g., A=SIN BC→A=SIN (B*C)

(2) Definition of subscripted variables (such as A( )) by using the DIM statement. On the PC-1245 Series, if, for example, DIM A(3Ø) is executed, memory locations for A(27) through A(3Ø) are set aside as an extension of a fixed variable definition area. On the PC-1460, however, the execution of DIM A(3Ø) reserves a separate memory area for array variables A(Ø) through A(3Ø) for the array named A. When defining subscripted variables (such as A( )) as an extension of fixed variables, use the specification on the right side of the following example:

DIM A(3Ø)→A(3Ø)=Ø

(3) Data I/O statement for tape files:
On the PC-1245 Series, the execution of, for instance, the PRINT# C statement saves the contents of the variable C and all the subsequent variables to a tape file. On the PC-1460, however, the execution of the same statement saves the contents of the variable C only. To save the contents of a specific variable and all the subsequent variables, use the specifications on the right side of the following examples:

e.g., PRINT#A→PRINT#A*
INPUT#C→INPUT#C*

(4) Value of a loop variable after completion of a FOR-NEXT loop:
The value of a loop variable obtained after the execution of a FOR-NEXT loop completed on the PC-1460 is different from that obtained on the PC-1245 Series. If the value of a loop variable is used in a conditional expression in a PC-1245 Series program, increment it by one when it is used on the PC-1460.

e.g., 1Ø FOR I=Ø TO 1Ø

5Ø NEXT I
6Ø IF I=1Ø THEN 1ØØ
Modify the value of I in line 6Ø as follows:
6Ø IF I=11 THEN 1ØØ
(On the PC-1460, the value of a loop variable must be incremented by one step value. The number of loop execution cycles remains the same, however.)

(5) Redefining [=]
The equal [=] key does not function as a definable key on the PC-1460. Accordingly, a different key should be used in programs in which the equal key is defined.

e.g., 1ØØ "=":... →1ØØ "N":...

(6) Exponent symbol "IE":
The PC-1460 uses the uppercase letter "E" for its exponent symbol. The following changes are required:

A=1.234 IE 5→A=1.234E5
B=IE 6→B=1E6

If a PC-1245 program is read from a tape file into the PC-1460, the change for the exponent symbol described just above will automatically be done by the PC-1460.

(7) The character codes of the PC-1245 Series are partially different from those of the PC-1460.
When the following codes are designated by the CHR$ function, change the codes.

Character Code	PC-1245	PC-1460
39 (&27)	⌐¬	'
91 (&5B)	√‾	[
92 (&5C)	¥	\
93 (&5D)	π	]
96 (&6Ø)	IE	‘
25Ø (&FA)	−(Error)	⌐¬
251 (&FB)	−(Error)	π
252 (&FC)	−(Error)	√‾

Note: As shown above, the PC-1460 does not have the character IE.

Additional modifications

(1) The PC-1245 Series uses a line number ranging from 1 to 999, whereas the PC-1460 has an extended line number ranging from 1 to 65279. Therefore, the line number uses 3 bytes in RAM (PC-1245 Series uses 2 bytes). The modification is carried out automatically when the program is loaded through the cassette tape. However, there is a possibility of memory overflow (ERROR 6) when loading or executing a long program. Further, when a single line is close to 80 bytes long, this modification may result in the erasing of the end of the line.

(2) If the tape stops or the read alarm stops when reading a program from a tape with PC-1245 Series programs, the computer will remain busy for 1 or 2 seconds and two asterisks will appear in the display.
This is because the computer is modifying the line numbers as described in (1) above.

Modifications Required for PC-1210 Series

To use PC-1210 Series programs on the PC-1460, they must be modified in the same way as PC-1245 Series programs except items (2) and (7). In addition, the following modifications are necessary.

(1) IF statement
If, for example,

```
5Ø IF A>L  PRINT "A" (display "A" if A>L)
```

is found in the program for the PC-1210 Series Pocket Computers, it is interpreted as

```
5Ø IF A>LPRINT "A" (Print out "A" if A>)
```

and results in an error when it is entered through the keyboard.
The error occurs because a command which does not exist in the PC-1210 Series does in fact exist in the PC-1460. To solve this problem, insert a THEN command into the IF statement as follows.

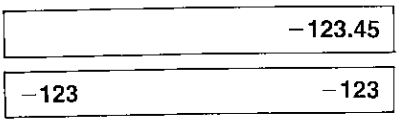
```
5Ø IF A>L THEN PRINT "A"
```

(2) Specified format in USING
The function of the USING command differs between the PC-1460 and the PC-1210 Series as follows.
Example:

```
10 A = −123.456
20 PAUSE USING "####.##"; A
30 PAUSE A, USING "####"; A
```

Executing this program displays the following.

*PC-1210/PC-1211



*PC-1460



For the execution of line 30 in the PC-1210 Series, the display on the left side also follows the displayed format on the right side. In the PC-1460, the display follows the previous specified format. This applies not only to the PAUSE command, but also to the PRINT and LPRINT commands.

(3) Omitting ")"
In the PC-1210 Series, the ")" which comes immediately before the [ENTER] or : (colon) can be omitted. It cannot be omitted in the PC-1460. Therefore, be sure to add the ")" to the program, if omitted.

(4) Print command
The PC-1460 has a PRINT command for displays and an LPRINT command for printing. However, all PRINT commands can be used for printing if PRINT = LPRINT is specified.
The PC-1210 Series does not have the LPRINT command. To print using a PC-1210 Series program, add PRINT = LPRINT to the program or, execute manually.

(5) Variables
When the RUN command has been executed in the PC-1210 Series, all variables are retained. In the PC-1460, however, all variables from A(27) and upwards are cleared.
Therefore, if there is a need to retain variables at the start of program execution, start the program execution using the GOTO command or function defined keys.

Modifications Required for PC-1260 Series

(1) Character Code modification
Character code 96 (&60) is a space in PC-1260 Series but is a left single quote (') in PC-1460. Accordingly, when the CHR$ command is used to specify a space with character code 96, change this code to character code 32 (&20).

(2) CLS, CURSOR commands
PC-1460 does not have the CLS or CURSOR display commands. Deletion and modification of these commands in any programs containing these are required.

Modifications Required for PC-1401, PC-1403, and PC-1450 Series

Programs written on PC-1401, PC-1403, and PC-1450 Series computers can be used without modification on the PC-1460.

Modifications Required for PC-1350 and PC-2500 Series

PC-1460 does not contain the following commands. Any program containing these must be modified.
CLS, CURSOR, MEM$, GCURSOR, GPRINT, LINE, POINT, PRESET, PSET, (TEST)

Programming Examples

Having read the description of each of the various functions in the preceding chapters, you have by now gained a knowledge of a number of program commands. However, in order for you to have a command of developing application programs in BASIC language, it is absolutely necessary that you write and execute your own practical application programs as well as those explained in this manual.

Just as you can improve your driving skill by actually operating the steering wheel or your tennis game by swinging the racket, proficiency in programming can only be attained by practicing as many programs as possible, regardless of the degree of your skill at each practice.

It is also very important for you to refer to programs developed by others. In this chapter, some programming examples using various commands in "BASIC" language are introduced to your reference.

For better understanding of the programming examples in this chapter, the conventions used in such examples are explained as follows:

① PROGRAM LIST
All the program lists contained in the programming examples are provided using the hard-copy outputs from the CE-126P or CE-140P printer in actual size or in 55% reduced size.

② PROGRAM CAPACITY
At the end of each program list, the capacity of the program itself is indicated in number of bytes.

③ PRINTOUT
For a program requiring a printout, the output of the program executed using the CE-126P or CE-140P printer is given in actual size or in 55% reduced size. A printout reduced to other than 55% is indicated with the different reduction rate used.

④ MEMORY CONTENTS
In the table of memory contents in each program example, variables with predetermined use are indicated by their specific use and those without predetermined use (e.g., those to be stored in the work area to retain intermediate results of a calculation, etc.) are indicated by the checkmark "√".

⑤ Then using CE-515P (or CE-516P)

In the table of contents of this chapter, "P" preceding a program title indicates that the CE-140P, CE-515P, or CE-516P may be used as a peripheral unit in executing the program. However, when using the CE-515P or CE-516P, observe the following points:

1. DIP switch setting
   - With CE-515P ..... Set all the DIP switch pins to the OFF position.
   - With CE-516P ..... Set the Nos. 1 to 5 switch pins to the OFF position and the No. 6 switch pin to the ON position.
2. Paper to be used
   Use a paper roll in either case.

NOTE: When the CE-515P or CE-516P is used, its hard-copy outputs differ partially from those of the CE-140P in the following:

(1) Color
CE-140P: purple → CE-515P (or CE-516P): blue

(2) Character
CE-140P: a → CE-515P: *a*
(CE-516P prints the same character as CE-140P.)

SHARP CORPORATION and/or its subsidiaries assume no responsibility or obligation for any financial losses or damages that may be incurred from using any of the examples of programs described in this manual. When using these programs, be aware that these programs may not fully satisfy your purpose or some programs may not be as precise as you wish them to be. Therefore, please carefully check the data in each of the program examples you use and confirm that they meet your requirements. If not, please modify them as required to meet your purpose before using them.

CONTENTS

Program Title	Page
• Conversions between Orthogonal Coordinates and Polar Coordinates	282
P• Exponential Regression Plot	287
P• Pareto Diagram	294
• Calculation of Area of N-sided Polygon	303
• A Circle Osculating Two Circles	310
P• Frequency Characteristic Graph	314
P• Circular Graph	321
• Transfer of Program File	332
P• Three-Dimensional Graph	337
• Number Guessing Game	341

PROGRAM TITLE: Conversions between Orthogonal Coordinates and Polar Coordinates

This is a very useful program for effecting conversions between orthogonal coordinates and polar (spherical) coordinates in three dimensions. When each data for conversion is input, the result of the conversion is obtained according to the unit of angle which is effective at that time.

■ HOW TO OPERATE

1. Press DEF A (for conversion to polar coordinate from orthogonal coordinates).
   When the value of each of orthogonal coordinates x, y, and z is input according to the display, the value of each of polar coordinates r, θ, and φ appears on the screen in the order named and then the program ends.
2. Press DEF B (for conversion to orthogonal coordinates from polar coordinates).
   When the value of each of polar coordinates R, θ, and φ is input, each value of x, y, and z appears on the screen in the order named and then the program ends.

Note: When the unit of angle specification is DEG, the result of a conversion is obtained in units of degrees. Likewise, when the angular unit specified is RAD, conversion is performed in units of radians.

■ REMARKS

1. Conversion to polar coordinates from orthogonal coordinates r=0 and θ=indefinite if r=y=0

   $r=\sqrt{x^2+y^2+z^2}$

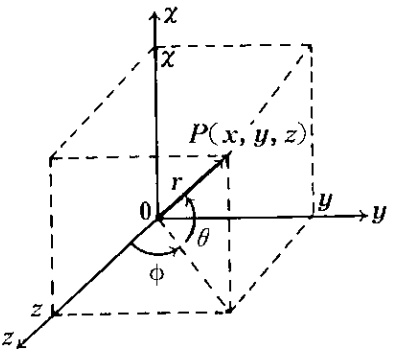
   $\theta=\sin^{-1}(z/r)$
   $\phi=\tan^{-1}(y/x)$ if $x>0$
   $\phi=90°$ if $x=0$ and $y\geq 0$
   $\phi=-90°$ if $x=0$ and $y<0$
   $\phi=\tan^{-1}(y/x)+180°$ if $x<0$ and $y\geq 0$
   $\phi=\tan^{-1}(y/x)-180°$ if $x<0$ and $y<0$

2. Conversion to orthogonal coordinates from polar coordinates

$x = r\cos\theta \cdot \cos\phi$

$y = r\cos\theta \cdot \sin\phi$

$z = r\sin\theta$



■ EXAMPLES

1. Convert orthogonal coordinates to polar coordiantes

   x=−1

   y=2

   z=−3

   Angular unit specification: DEG

2. Convert polar coordinates to orthogonal coordinates

   r=3.741657387

   A=−53.30077479

   C=116.5650512

   Angular unit specification: DEG

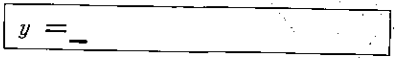
■ KEY OPERATION SEQUENCE

[Orthogonal coordinates → Polar coordinates]

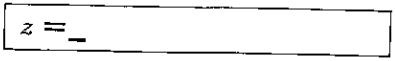
1. DEF A



2. −1 ENTER



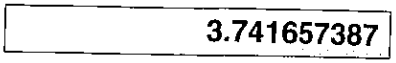
3. 2 ENTER



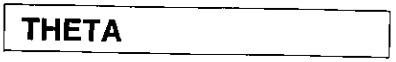
4. −3 ENTER



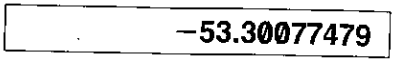
5. ENTER



6. ENTER



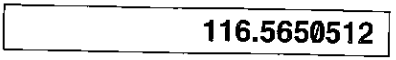
7. ENTER



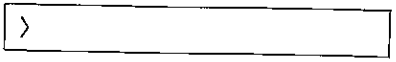
8. ENTER



9. ENTER

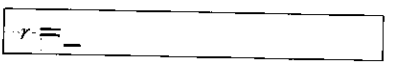


10. ENTER

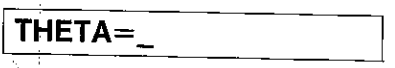


[Polar coordinates → Orthogonal coordinates]

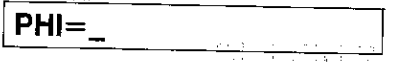
1. DEF B



2. 3.741657387 ENTER



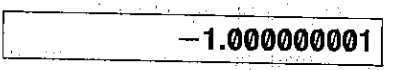
3. −53.30077479 ENTER



4. 116.5650512 ENTER



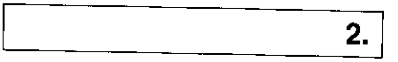
5. ENTER



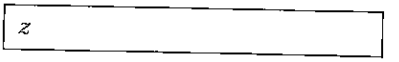
6. ENTER



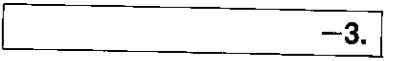
7. ENTER



8. ENTER



9. ENTER



10. ENTER



■ PROGRAM LIST

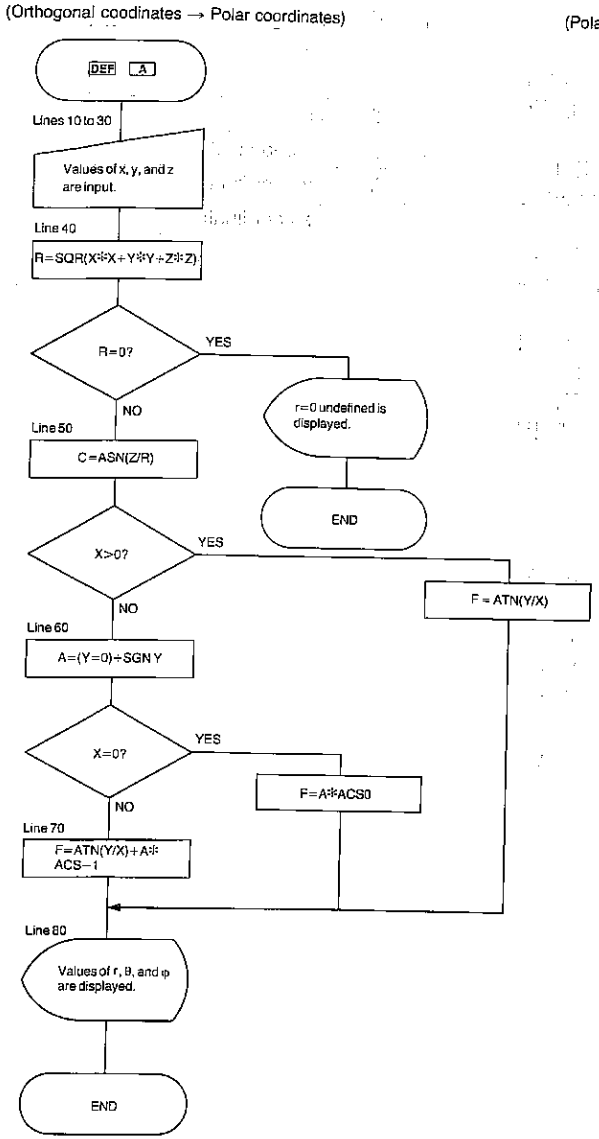
```
10:"A":INPUT "x=";X
20:INPUT "y=";Y
30:INPUT "z=";Z
40:R=SQR (X*X+Y*Y+Z*Z):
   IF R=0 PAUSE "r=0 Un
   defined":END
50:C=ASN (Z/R):IF X>0
   LET F=ATN (Y/X):GOTO
   80
60:A=(Y=0)+SGN Y:IF X=0
   LET F=A*ACS 0:GOTO 8
   0
70:F=ATN (Y/X)+A*ACS -1
80:WAIT :PRINT "r":
   PRINT R:PRINT "THETA
   ":PRINT C:PRINT "PHI
   ":PRINT F:END
90:"B":INPUT "r=";R
100:INPUT "THETA=";C
110:INPUT "PHI=";F
120:Y=R*COS C:X=Y*COS F:
   Y=Y*SIN F:Z=R*SIN C
130:WAIT :PRINT "x":
   PRINT X:PRINT "y":
   PRINT Y:PRINT "z":
   PRINT Z:END
```

300 Bytes

■ MEMORY CONTENTS

A	$\sqrt{}$
C	θ
F	ϕ
R	r
X	x-coordinate
Y	y-coordinate
Z	z-coordinate

■ FLOWCHARTS


(Orthogonal coodinates → Polar coordinates)
DEF A
Lines 10 to 30
Values of x, y, and z are input.
Line 40
R=SQR(X*X+Y*Y+Z*Z)
R=0?
YES
r=0 undefined is displayed.
END
NO
Line 50
C=ASN(Z/R)
X>=0?
YES
F=ATN(Y/X)
NO
Line 60
A=(Y=0)÷SGN Y
X=0?
YES
F=A*ACS0
NO
Line 70
F=ATN(Y/X)+A*ACS-1
Line 80
Values of r, θ, and φ are displayed.
END
(Polar coordinates → Orthogonal coordinates)
DEF B
Lines 90 to 110
Values of r, θ, and φ are input.
Line 120
Y=R*COS C
X=Y*COS F
Y=Y*SIN F
Z=R*SIN C
Line 130
Values of x, y, and z are displayed.
END.

PROGRAM TITLE: Exponential Regression Plot

Required Peripheral Equipment
CE-140P or CE-515P (or CE-516P) and CE-516L

In the course of Nature, a great number of changes take place in the manner of exponential functions as you will find in the relationship between time and temperature when something hot gets cold, the relationship between time and disintegration rate of atomic nucleus, etc. This program solves for the values of coefficients a and b and coefficient of correlation r by applying data inputs x and y to exponential curve $y=a \cdot b^x$. Then, the exponential curve is output on the printer with the input data and estimated values plotted on the graph.

■ HOW TO OPERATE

1. Press [DEF] [A]. (Program starts)
2. Input the number of data (N) according to the display. When data x and y are input repeatedly as specified by the number of data (N), the values of coefficients a and b and coefficient of correlation r are calculated and are output on the printer.
3. When keys [DEF] [B] are operated, the exponential curve is output on the printer in green color and the input data x's and y's are plotted on the graph in purple color.
4. Next, when you input the estimated values of x's, the input data are plotted on the graph in red color and the estimated values of x's and y's are printed out. Then, the program ends.

Note: The number of input data must be within the following limit:
CE-212M (or CE-201M): approx. 180
CE-2H16M (or CE-202M): 255

■ REMARKS

An infinite (N) number of points given (x_1, y_1), (x_2, y_2) ... (x_n, y_n) are applied to exponential function $y=a \cdot b^x$. This function is substituted with $Y=l_n y$, $A=l_n a$, $B=l_n b$ and transformed into linear expression $Y=A+Bx$. By the method of least square, A and B are then determined as follows:

$$B=\frac{\sum x_i \cdot Y_i - n \cdot \bar{x} \cdot \bar{Y}}{\sum x_i^2 - n \cdot \bar{x}^2}, \quad A=\bar{Y}-B \cdot \bar{x}$$

where: $Y_i=l_n y_i$, $\bar{Y}=\frac{1}{n}\sum_{i=1}^{n} Y_i$, $\bar{x}=\frac{1}{n}\sum_{i=1}^{n} x_i$

Once A and B have been determined, a and b can be obtained from $a=e^A$, $b=e^B$, as well as the coefficient of correlation r between x and y.

■ EXAMPLE

x	0.5	1.2	3.1	7.4
y	7.01	11.72	44.54	936.71

N=4

Apply the above data to $y=a \cdot b^x$ and then estimate the values of y's with the estimated values of x's taken as 2, 4, 6, and 6.5, respectively.

■ When using CE-515P (or CE-516P)

Modify lines 130 and 150 in the program list on page ; and add line 375 to the list as follows:

```
130:GOSUB 375

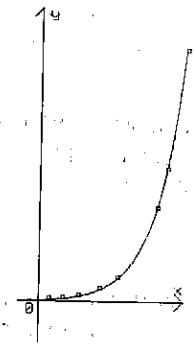
150:"B":N=F/300: GOSUB 375

375:CLOSE : OPEN "1200,N,8,1,A,L"
```

■ PRINTOUT

(Printed out in color.)

```
r=9.999942365E-01
a=4.960331916
b=2.03057723
```




```
* Estimation *
x=2          y=20.45265825
x=4          y=84.3312981
x=6          y=347.7185094
x=6.5        y=495.4830476
```

■ KEY OPERATION SEQUENCE

<Data input>

1. DEF A

N=_

2. 4 ENTER

x (1) =

?

3. 0.5 ENTER

y (1) =

?

4. 7.01 ENTER

x (2) =

?

5. 1.2 ENTER

y (2) =

?

6. 11.72 ENTER

x (3) =

?

⋮

Input data in the same manner as above.

⋮

7. 936.71 ENTER

〉

<Output of operation results>

1. DEF B

Est. x=_

2. 2 ENTER

Est. x=_

3. 4 ENTER

Est. x=_

4. 6 ENTER

Est. x=_

5. 6.5 ENTER

〉

■ PROGRAM LIST

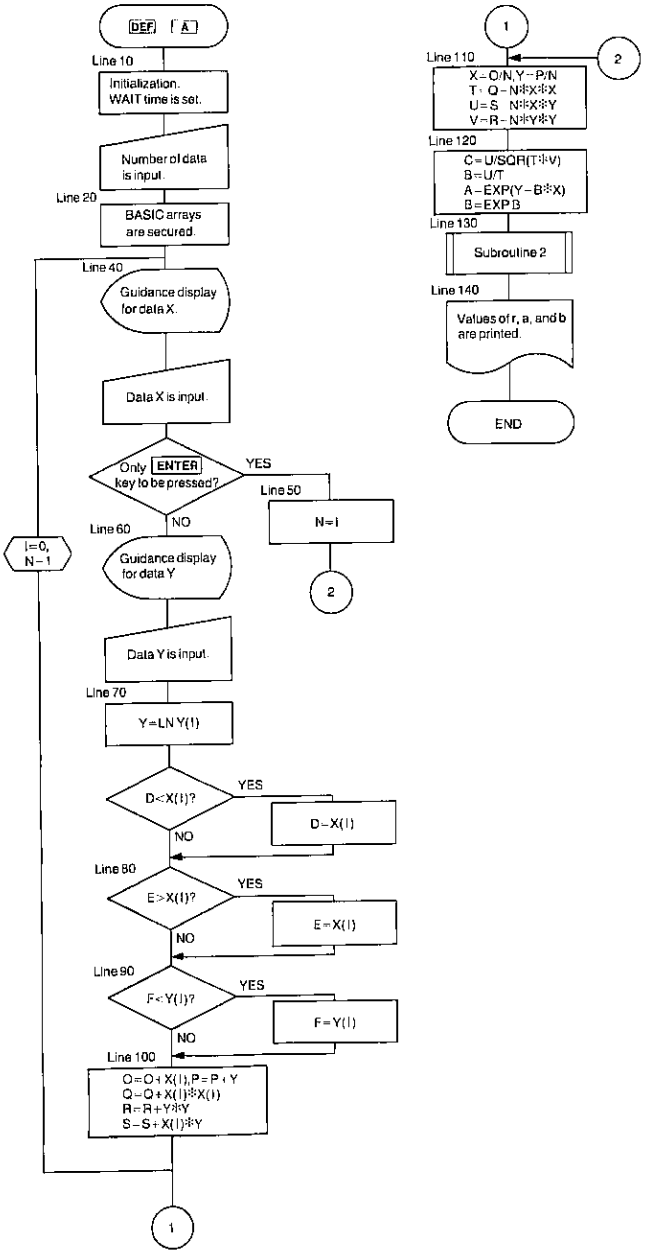
```
 10:"A": CLEAR : WAIT 0: INPUT "N=";N
 20:DIM X(N-1),Y(N-1),XS(N-1),YS(N-1):E
    =10^8:D=-E:F=D
 30:FOR I=0 TO N-1
 40:PAUSE "x("+STR$ (I+1)+")=": INPUT X
    (I): GOTO 60
 50:N=I: GOTO 110
 60:PAUSE "y("+STR$ (I+1)+")=": INPUT Y
    (I)
 70:Y=LN Y(I): IF D<X(I) LET D=X(I)
 80:IF E>X(I) LET E=X(I)
 90:IF F<Y(I) LET F=Y(I)
100:O=O+X(I):P=P+Y:Q=Q+X(I)*X(I):R=R+Y*
    Y:S=S+X(I)*Y: NEXT I
110:X=O/N:Y=P/N:T=Q-N*X*X:U=S-N*X*Y:V=R
    -N*Y*Y
120:C=U/SQR (T*V):B=U/T:A=EXP (Y-B*X):B
    =EXP B
130:GOSUB 380
140:LPRINT "r=";C: LPRINT "a=";A:
    LPRINT "b=";B: END
150:"B":M=F/300: GOSUB 380
160:IF E>=0 LET Z=25:L=D/175: GOTO 180
170:L=(D+ABS E)/200:Z=ABS E/L+5
180:GRAPH : GLCURSOR (Z+230,-225)
185:SORGN
190:LLINE (-Z,0)-(200-Z,0)-(200-Z-10,-1
    0),0
200:LLINE (200-Z,0)-(200-Z-10,10):
    LPRINT "Px"
210:LLINE (0,-50)-(0,350)-(-10,340)-(0,
    350)
215:LLINE -(10,340): LPRINT "Py"
220:GLCURSOR (-15,-15): LPRINT "P0"
230:COLOR 1: FOR I=0 TO N-1:J=X(I)/L:K=
    Y(I)/M: GOSUB 370: NEXT I
240:COLOR 2:J=-Z
245:GLCURSOR (J,A*B^(J*L)/M)
250:J=J+2: IF J>200-Z GOTO 270
260:K=A*B^(J*L)/M: IF K>350 THEN 270
265:LLINE -(J,K): GOTO 250
270:I=0: COLOR 3
280:INPUT "Est. x=";XS(I): GOTO 300
290:GOTO 320
300:J=XS(I)/L:YS(I)=A*B^XS(I):K=YS(I)/M
    : IF K<=350 GOSUB 370
310:I=I+1: IF I<N THEN 280
320:GLCURSOR (0,-50)
322:LTEXT : LPRINT
325:COLOR 0: IF I=0 THEN 350
330:LPRINT "* Estimation *"
340:FOR H=0 TO I-1: LPRINT USING "&&&&&
    &&&&&&&&&&&&&";"x="+STR$ XS(H);
350:LPRINT "y="+STR$ YS(H): NEXT H
360:USING : LF 5: END
370:GLCURSOR (J,K): LPRINT "P0": RETURN
380:CONSOLE 160: LTEXT : LPRINT
390:CSIZE 2: COLOR 0: RETURN
```

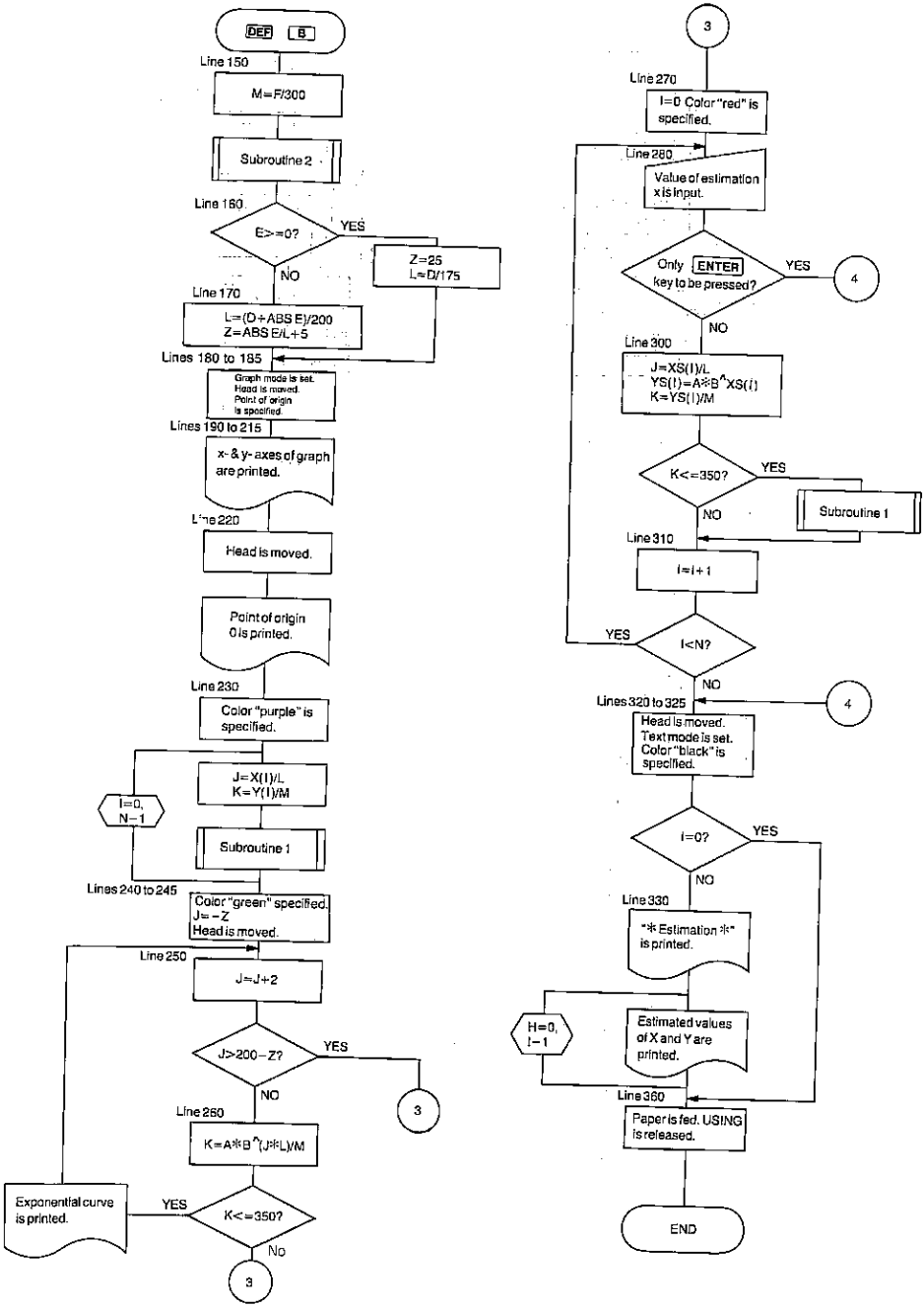
1077 Bytes

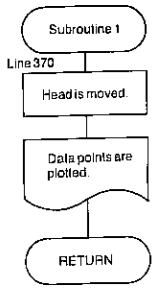
■ MEMORY CONTENTS

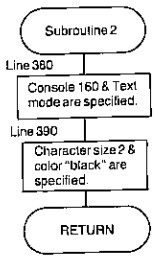
A	a
B	b, b
C	Correlation of coefficient r
D	X-MAX.
E	X-MIN.
F	Y-MAX.
H	√
I	√
J	√
K	√
L	√
M	√
N	Number of data
O	Σxi
P	ΣYi
Q	$\Sigma x^2 i$
R	$\Sigma Y^2 i$
S	Σxi · Yi
T	Sxx
U	Sxy
V	Syy
X	$\overline{X}$
Y	ln yi,$\overline{Y}$
Z	√
X(N-1)	Data X
XS(N-1)	Estimation X
Y(N-1)	Data Y
YS(N-1)	Estimation Y

■ FLOWCHARTS


DEF FN A
Line 10
Initialization.
WAIT time is set.
Number of data is input.
Line 20
BASIC arrays are secured.
Line 40
Guidance display for data X.
Data X is input.
Only ENTER key to be pressed?
YES
Line 50
N=I
2
NO
Line 60
Guidance display for data Y
I=0, N-1
Data Y is input.
Line 70
Y=LN Y(I)
D<X(I)?
YES
D=X(I)
NO
Line 80
E>X(I)?
YES
E=X(I)
NO
Line 90
F<Y(I)?
YES
F=Y(I)
NO
Line 100
O=O+X(I),P=P+Y
Q=Q+X(I)*X(I)
R=R+Y*Y
S=S+X(I)*Y
1
1
2
Line 110
X=O/N,Y=P/N
T=Q-N*X*X
U=S-N*X*Y
V=R-N*Y*Y
Line 120
C=U/SQR(T*V)
B=U/T
A=EXP(Y-B*X)
B=EXP B
Line 130
Subroutine 2
Line 140
Values of r, a, and b are printed.
END


DEF B
Line 150
M=F/300
Subroutine 2
Line 160
E>=0?
YES
NO
Z=25
L=D/175
Line 170
L=(D+ABS E)/200
Z=ABS E/L+5
Lines 180 to 185
Graph mode is set.
Head is moved.
Point of origin is specified.
Lines 190 to 215
x- & y- axes of graph are printed.
Line 220
Head is moved.
Point of origin 0 is printed.
Line 230
Color "purple" is specified.
J=X(I)/L
K=Y(I)/M
I=0, N-1
Subroutine 1
Lines 240 to 245
Color "green" specified.
J=-Z
Head is moved.
Line 250
J=J+2
J>200-Z?
YES
NO
3
Line 260
K=A*B^(J*L)/M
Exponential curve is printed.
YES
K<=350?
No
3
3
Line 270
I=0 Color "red" is specified.
Line 280
Value of estimation x is input.
Only ENTER key to be pressed?
YES
4
NO
Line 300
J=XS(I)/L
YS(I)=A*B^XS(I)
K=YS(I)/M
K<=350?
YES
NO
Subroutine 1
Line 310
I=I+1
YES
I<N?
NO
4
Lines 320 to 325
Head is moved.
Text mode is set.
Color "black" is specified.
I=0?
YES
NO
Line 330
"* Estimation *" is printed.
H=0, I-1
Estimated values of X and Y are printed.
Line 360
Paper is fed. USING is released.
END


Subroutine 1
Line 370
Head is moved.
Data points are plotted.
RETURN


Subroutine 2
Line 380
Console 160 & Text mode are specified.
Line 390
Character size 2 & color "black" are specified.
RETURN

PROGRAM TITLE: Pareto Diagram

Required Peripheral Equipment
CE-140P or CE-515P
(or CE-516P) and CE-516L

The basic method of conducting quality control is to first determine the degree of significance of each of various quality control factors and then tackling the problems with higher significance. Many points, such as labor, cost, productivity, etc., have a bearing on factors for controlling quality. Pareto diagram is a convenient diagram to find out which factors are more important than the others. This program has been designed to print out a simple, easy-to-see diagram on which various factors are arranged in the order of the largest number of data.

■ HOW TO OPERATE

1. Press [DEF] [A] . (Program starts)
2. Input the factor name (within 8 characters) along with the number of defects (within 9 characters) according to the display. Repeat this step for each factor as required for the number of factors.
   If you press [ENTER] only while the computer is in the wait state for the input of a factor name, you can proceed to step 3 below.
3. Determine whether or not you need to correct the contents of any input.
   - When to be corrected — Press [Y] [ENTER] .
   - When not to be corrected — Press [N] [ENTER] .

   (In this case, you can proceed to step 4 below.)
   Method of data correction
   A factor name is first displayed. If the factor name is to be corrected, input the correct factor name. If not to be corrected, press [ENTER] only. The same procedure applies to the correction of data (i.e., number of defects). If you repeat this step for the number of factors you have entered, then you must again determine whether or not you need to correct the contents of any input.
4. Input the title name (within 10 characters), and the date of compilation (or drawing) according to the display. A table containing the input data (factor names and number of defects) and the component ratio of data classified by factor and a Pareto diagram are output on the printer. Then, the program ends.

Note: The number of factors can be increased by modifying the value of A=8 in line 10 of the program list on page 000.
CE-212M (or CE-201M) approx. 139 factors
CE-2H16M (or CE-202M) 255 factors

■ EXAMPLE

Shown below is a table indicating the number of defective products that occurred from each machine at the factory of a major electrical home appliance manufacturer. Prepare a Pareto diagram using the date of compilation as March 31, 1986, and the title name as "Pareto".

No.	Factor name	Data
1	MACHINE A	10
2	MACHINE B	20
3	MACHINE C	15
4	MACHINE D	30
5	MACHINE E	40
6	MACHINE F	80
7	MACHINE G	100
8	MACHINE H	90

■ When using CE-515P (or CE-516P)

Modify line 320 of the program list on page 297 as follows:

```
320:CONSOLE 80: CLOSE : OPEN "L200,N,8,
    1,A,L": GOSUB 620:H=0:U=0: GOSUB 71
    0
```

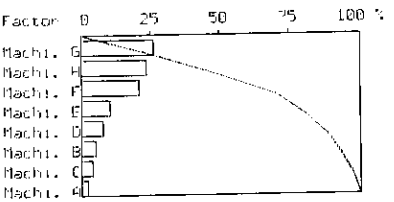
■ PRINTOUT

(Printed out in color. Refer to page I.)

Pareto

DMG date: 1986/3/31

No.	Factor	Defect	Ratio %
1	Machi. G	100	25.97
2	Machi. H	90	23.37
3	Machi. F	80	20.77
4	Machi. E	40	10.38
5	Machi. D	30	7.79
6	Machi. B	20	5.19
7	Machi. C	15	3.89
8	Machi. A	10	2.59



■ KEY OPERATION SEQUENCE

1. DEF A

```
*Pareto Diagram*
Fact. name (1)
=_
```

2. M SML ACHI. SML SPC A ENTER

```
No. of def. (1)
=_
```

3. 10 ENTER

```
Fact. name (2)
=_
```

⋮

Input data in the same manner as above.

⋮

4. 90 ENTER

```
Correct? (Y/N)_
```

5. Y ENTER

```
(1)Machi. A
—> _
```

6. ENTER

```
10
—> _
```

7. ENTER

```
(2)Machi. B
—> _
```

8. ENTER

```
15
—> _
```

9. 20 ENTER

```
(3)Machi. C
—> _
```

⋮

Confirm the data and correct it if necessary in the same manner as above.

⋮

10. ENTER

```
Correct? (Y/N)_
```

11. N ENTER

```
Title=_
```

12. P SML ARETO ENTER

```
Year=19_
```

13. 86 ENTER

```
Month=_
```

14. 3 ENTER

```
Day=_
```

15. 31 ENTER

```
>
```

■ PROGRAM LIST

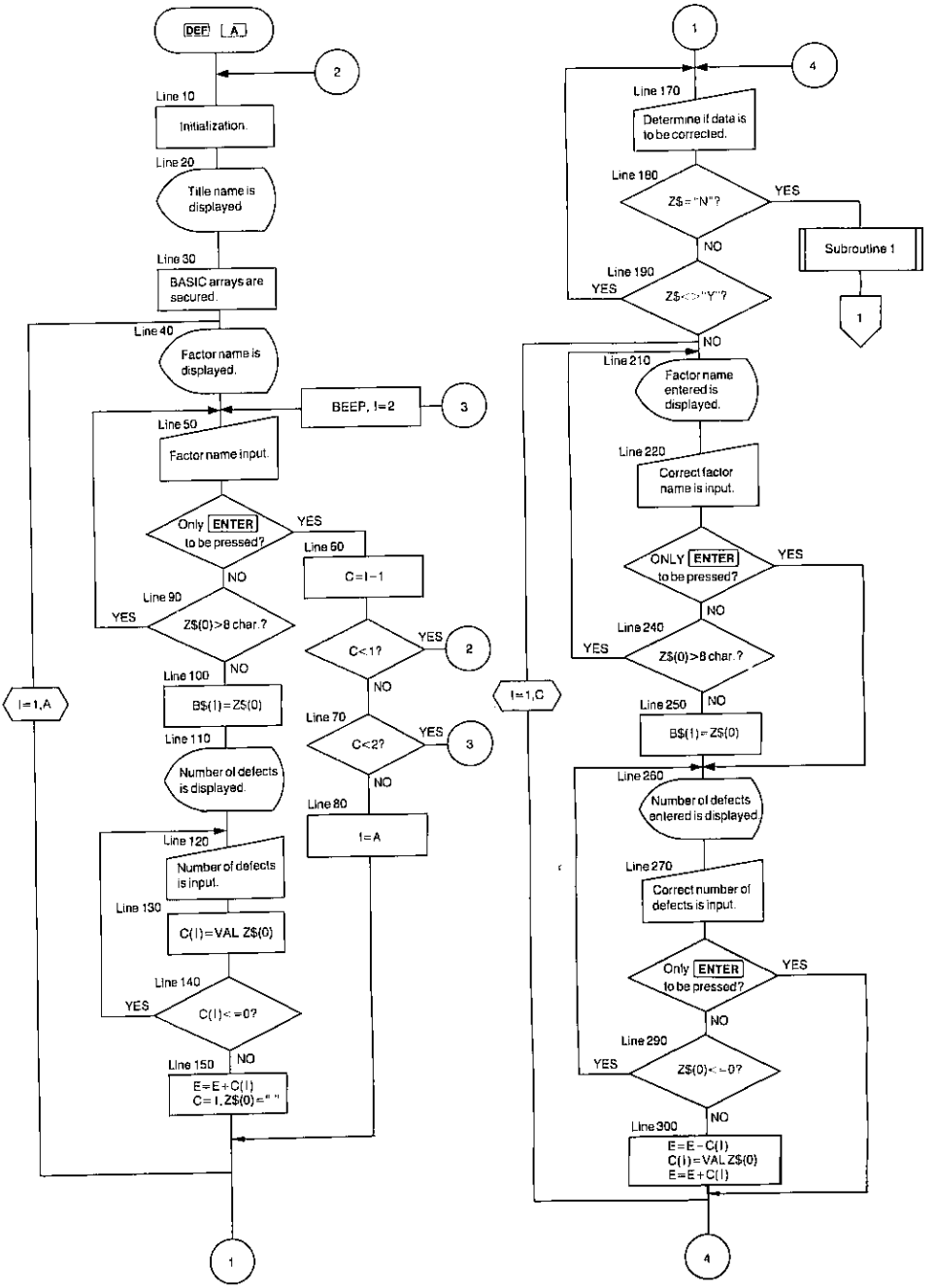
```
10:"A": USING : CLEAR : WAIT 0:A=8:B=8
20:PAUSE "<Pareto Diagram>"
30:DIM B$(A),KB,C(A),D(A),Z$(0)*18,N$(0
   )*40
40:FOR I=1 TO A: PAUSE "Fact. name (";
   STR$ I;")"
50:BEEP 1: INPUT "=";Z$(0): GOTO 90
60:C=I-1: IF C<1 BEEP 2: GOTO 10
70:IF C<2 BEEP 2:I=2: GOTO 50
80:I=A: GOTO 160
90:IF LEN Z$(0)>8 THEN 50
100:B$(I)=Z$(0)
110:PAUSE "No.of def. (";STR$ I;")"
120:BEEP 1: INPUT "=";Z$(0)
130:C(I)=VAL Z$(0)
140:IF C(I)<=0 BEEP 2: GOTO 120
150:E=E+C(I):C=I:Z$(0)=""
160:NEXT I
170:BEEP 1: INPUT "Correct? (Y/N)";Z$:Z
   $=CHR$ (ASC Z$ AND 223)
180:IF Z$="N" GOSUB 500: GOTO 320
190:IF Z$<>"Y" THEN 170
200:FOR I=1 TO C
210:WAIT 150: PRINT "(";STR$ I;")";B$(I
   )
220:INPUT " -> ";Z$(0): GOTO 240
230:GOTO 260
240:IF LEN Z$(0)>8 BEEP 2: GOTO 210
250:B$(I)=Z$(0)
260:PRINT STR$ C(I)
270:INPUT " -> ";Z$(0): GOTO 290
280:GOTO 310
290:IF VAL Z$(0)<=0 BEEP 1: GOTO 260
300:E=E-C(I):C(I)=VAL Z$(0):E=E+C(I)
310:NEXT I: GOTO 170
320:CONSOLE 80: GOSUB 620:H=0:U=0:
   GOSUB 710
330:LF 1: LTEXT : LPRINT
335:CSIZE 2
340:H=0:U=0: LF 2
350:LPRINT "Factor  0      25      50
    75    100 %"
360:GRAPH : GLCURSOR (95,10)
365:SORGN
370:LLINE -(340,0)-(340,C*-24)-(0,C*-24
   )-(0,0)
380:FOR I=1 TO 3
385:GLCURSOR (84*I,0)
390:LLINE -(84*I,5): NEXT I
400:GLCURSOR (0,-6)
405:COLOR 3
410:O=-6:P=-24: FOR I=1 TO C
420:LLINE -((D(I)-U)*3.4,O)-((D(I)-U)*3
   .4,P)-(0,P)
430:O=O-24:P=O-18
440:GLCURSOR (0,O):U=D(I): NEXT I
450:GLCURSOR (0,0)
455:COLOR 2
460:FOR I=1 TO C: LLINE -(D(I)*3.4,1*-2
   4): NEXT I
470:GLCURSOR (0,0)
475:COLOR 0: LTEXT : LPRINT
480:FOR I=1 TO C: LPRINT USING "&&&&&&&
   &";B$(I): NEXT I
490:LF 5: USING : END
500:L= INT (C/2+1):R=C
510:IF L>1 LET L=L-1:X=C(L):B$(0)=B$(L)
   : GOTO 540
520:X=C(R):B$(0)=B$(R):C(R)=C(L):B$(R)=
   B$(1):R=R-1
530:IF R<=1 LET C(1)=X:B$(1)=B$(0):
   GOTO 610
540:J=L
550:I=J:J=2*J
560:IF J>R THEN 600
570:IF J<R THEN IF C(J)>C(J+1) LET J=J+
   1
580:IF X<=C(J) THEN 600
590:C(I)=C(J):B$(I)=B$(J): GOTO 550
600:C(I)=X:B$(I)=B$(0): GOTO 510
610:FOR I=1 TO C:H=H+C(I):D(I)=H/E*100:
   NEXT I: RETURN
620:INPUT "Title=";N$(0)
630:IF LEN N$(0)>10 BEEP 2: GOTO 620
640:CC$="99": INPUT "Year =19";CC$
650:N=1900+ INT VAL CC$: IF N>1999 BEEP
   1: GOTO 640
660:DD$="99": INPUT "Month =";DD$
670:D= INT VAL DD$: IF D>12 BEEP 2:
   GOTO 660
680:EE$="99": INPUT "Day =";EE$
690:F1= INT VAL EE$: IF F1>31 THEN 680
700:RETURN
710:LTEXT : LPRINT
720:LF 1: CSIZE 3: GRAPH
725:COLOR 0
730:GLCURSOR (100,0): LPRINT "P";LEFT$
   (N$(0),10)
740:CSIZE 2: GLCURSOR (220,-80)
750:LPRINT "PDWG date: ";STR$ N;"/";
   STR$ D;"/";STR$ F1
760:GLCURSOR (10,-100)
765:SORGN
770:F=1: FOR I=0 TO C+1
780:IF F=-1 LET K=0: GOTO 800
790:K=444
800:LLINE -(K,G),0
810:GLCURSOR (K,G-30):F=F*-1:G=G-30:
   NEXT I
815:GLCURSOR (0,-(C+1)*30)
820:RESTORE :M=0:F=1: FOR I=1 TO 5
830:IF F=1 LET O=0: GOTO 850
840:O=-(C+1)*30
850:LLINE -(M,O),0: READ M
860:GLCURSOR (M,O):F=F*-1: NEXT I
870:GLCURSOR (12,-20): LPRINT "P";"No."
880:GLCURSOR (80,-20): LPRINT "PFactor"
890:GLCURSOR (200,-20): LPRINT "PDefect
   "
900:GLCURSOR (350,-20): LPRINT "PRatio
   %"
910:FOR I=1 TO C
920:GLCURSOR (10,1*-30-20): LPRINT "P";
   USING "###";I
930:GLCURSOR (70,1*-30-20)
940:USING : LPRINT "P";LEFT$ (B$(I),8);
   GLCURSOR (180,-1*30-20)
950:LPRINT "P"; USING "##########";C(I)
960:GLCURSOR (340,1*-30-20): LPRINT
   USING "####.##";"P";D(I)-U; USING :
   U=D(I)
970:NEXT I: RETURN
980:DATA 55,185,335,444,0
```

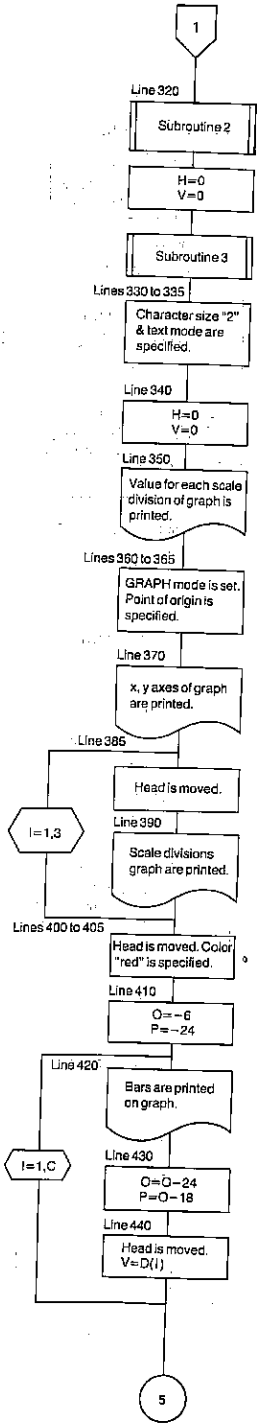
2218 Bytes

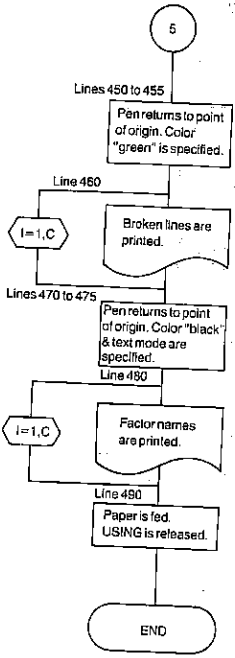
■ MEMORY CONTENTS

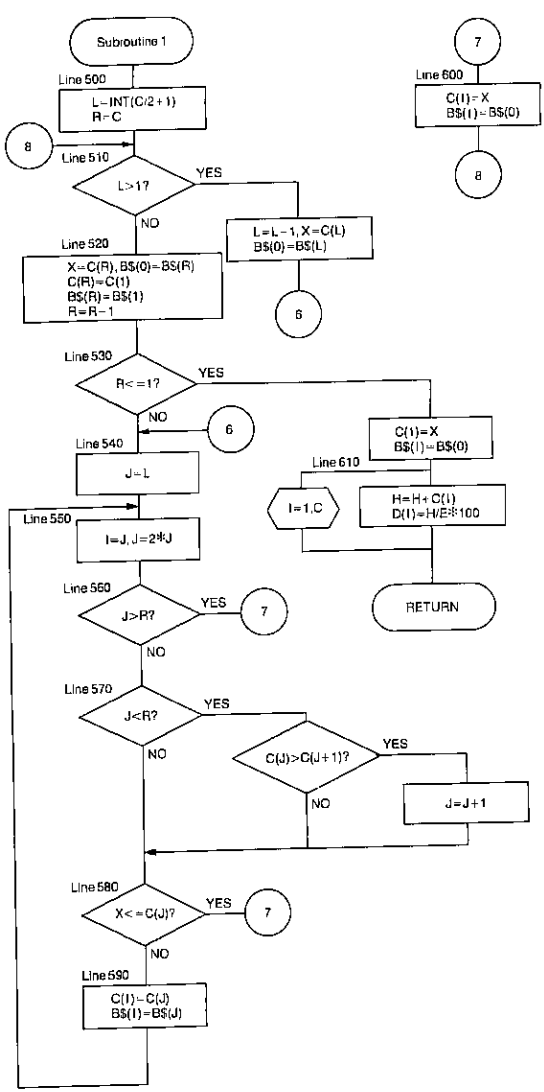
A	Number of factors
B	√
C	√
D	Month
E	Total
F	Flag
H	√
I	√
J	√
L	√
N	Year
R	End pointer
V	√
W	For data read
X	√
C (A)	Number of defects
D (A)	Component ratio
F1	Day
Z$	√
N$ (0)	Title name
Z$ (0)	For input
B$ (A)	Factor name
CC$	For input of year
DD$	For input of month
EE$	For input of day

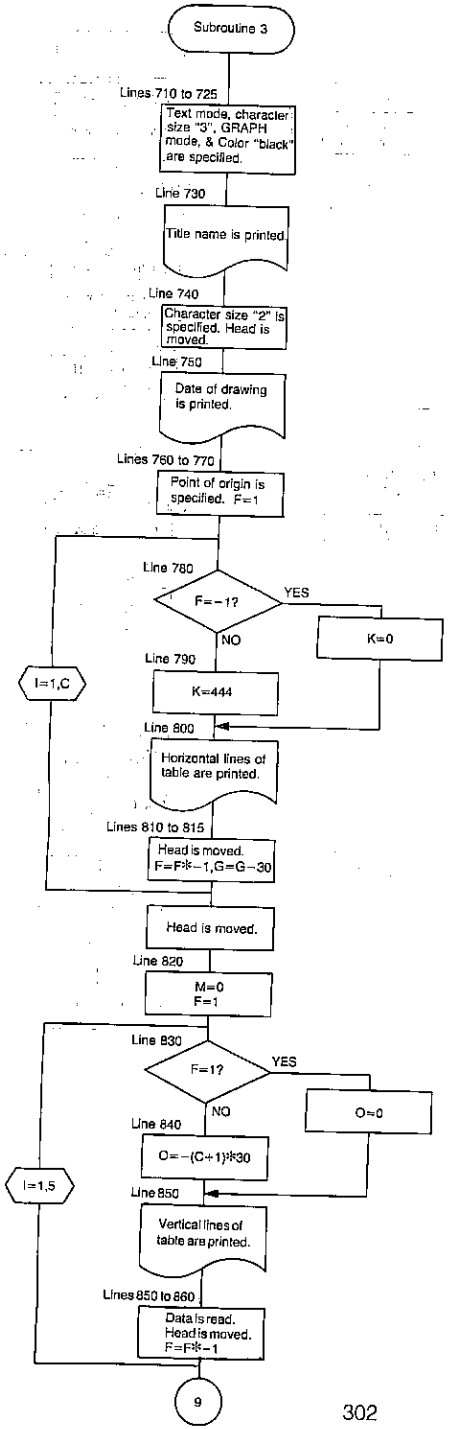
■ FLOWCHARTS


DEF A
2
Line 10
Initialization.
Line 20
Title name is displayed
Line 30
BASIC arrays are secured.
Line 40
Factor name is displayed.
BEEP, I=2
3
Line 50
Factor name input.
Only ENTER to be pressed?
YES
NO
Line 60
C=I-1
Line 90
YES
Z$(0)>8 char.?
NO
C<1?
YES
2
NO
Line 100
B$(1)=Z$(0)
Line 70
C<2?
YES
3
NO
Line 110
Number of defects is displayed.
Line 80
I=A
I=1,A
Line 120
Number of defects is input.
Line 130
C(I)=VAL Z$(0)
Line 140
YES
C(I)<=0?
Line 150
NO
E=E+C(I)
C=I, Z$(0)=" "
1
1
4
Line 170
Determine if data is to be corrected.
Line 180
Z$="N"?
YES
NO
Subroutine 1
1
Line 190
YES
Z$<>"Y"?
NO
Line 210
Factor name entered is displayed.
Line 220
Correct factor name is input.
ONLY ENTER to be pressed?
YES
NO
Line 240
YES
Z$(0)>8 char.?
I=1,C
Line 250
NO
B$(1)=Z$(0)
Line 260
Number of defects entered is displayed.
Line 270
Correct number of defects is input.
Only ENTER to be pressed?
YES
NO
Line 290
YES
Z$(0)<=0?
NO
Line 300
E=E-C(I)
C(I)=VAL Z$(0)
E=E+C(I)
4


1
Line 320
Subroutine 2
H=0
V=0
Subroutine 3
Lines 330 to 335
Character size "2" & text mode are specified.
Line 340
H=0
V=0
Line 350
Value for each scale division of graph is printed.
Lines 360 to 365
GRAPH mode is set. Point of origin is specified.
Line 370
x, y axes of graph are printed.
Line 385
Head is moved.
I=1,3
Line 390
Scale divisions graph are printed.
Lines 400 to 405
Head is moved. Color "red" is specified.
Line 410
O=−6
P=−24
Line 420
Bars are printed on graph.
I=1,C
Line 430
O=O−24
P=O−18
Line 440
Head is moved.
V=D(I)
5


5
Lines 450 to 455
Pen returns to point of origin. Color "green" is specified.
Line 460
I=1,C
Broken lines are printed.
Lines 470 to 475
Pen returns to point of origin. Color "black" & text mode are specified.
Line 480
I=1,C
Factor names are printed.
Line 490
Paper is fed. USING is released.
END


Subroutine 1
Line 500
L=INT(C/2+1)
R=C
8
Line 510
L>1?
YES
NO
L=L-1, X=C(L)
B$(0)=B$(L)
Line 520
X=C(R), B$(0)=B$(R)
C(R)=C(1)
B$(R)=B$(1)
R=R-1
6
Line 530
R<=1?
YES
NO
6
Line 540
J=L
Line 550
I=J, J=2*J
Line 560
J>R?
YES
7
NO
Line 570
J<R?
YES
NO
C(J)>C(J+1)?
YES
NO
J=J+1
Line 580
X<=C(J)?
YES
7
NO
Line 590
C(I)=C(J)
B$(I)=B$(J)
7
Line 600
C(I)=X
B$(I)=B$(0)
8
Line 610
I=1,C
H=H+C(I)
D(I)=H/E*100
RETURN
Subroutine 2
Line 620
Title name is input.
Line 630
N$(0)>10 char.?
YES
NO
Line 640
Year of drawing is input.
Line 650
N=1900+INT VAL CC$
N>1,999?
YES
NO
Line 660
Month of drawing is input.
Line 670
D=INT VAL DD$
D>12?
YES
NO
Line 680
Day of drawing is input.
Line 690
F1=INT VAL EE$
F1>31?
YES
NO
RETURN


Subroutine 3
Lines 710 to 725
Text mode, character size "3", GRAPH mode, & Color "black" are specified.
Line 730
Title name is printed.
Line 740
Character size "2" is specified. Head is moved.
Line 750
Date of drawing is printed.
Lines 760 to 770
Point of origin is specified. F=1
Line 780
F=−1?
YES
NO
K=0
I=1,C
Line 790
K=444
Line 800
Horizontal lines of table are printed.
Lines 810 to 815
Head is moved. F=F*−1,G=G−30
Head is moved.
Line 820
M=0
F=1
Line 830
F=1?
YES
NO
O=0
Line 840
O=−(C+1)*30
I=1,5
Line 850
Vertical lines of table are printed.
Lines 850 to 860
Data is read. Head is moved. F=F*−1
9
9
Lines 870 to 900
Head is moved.
Boxheads (No., Factor, Defect, Ratio %) are printed.
Line 920
Head is moved.
Item Nos. are printed.
Line 930
Head is moved.
Line 940
Factor names are printed.
Head is moved.
Lines 950 to 960
Number of defects and component ratio for each factor are printed.
USING is released. V=D(I)
I=1,C
RETURN

PROGRAM TITLE: Calculation of Area of N-sided Polygon

Required Peripheral Equipment
CE-126P

Any polygon is theoretically an aggregation of triangles. By utilizing this theory, let us calculate the area of a polygon. This program figures out the area of a polygon by dividing the polygon into triangles, calculating the area of each triangle, and obtaining the sum total of the areas of all the triangles.

■ HOW TO OPERATE

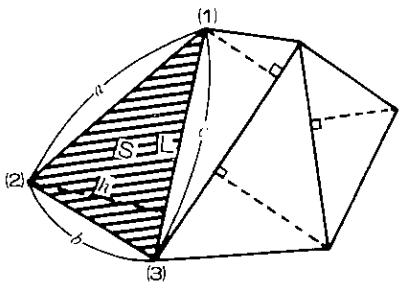
1. Press DEF A . (Program starts)
   Input the number of vertexes (i.e., points) and the coordinates x and y of each vertex according to the display.
2. Next, press DEF B .
   Input the vertex number (i.e., point number) of each of the three vertexes of the triangle according to the display, and the area of the triangle will be printed out. When you press ENTER before entering the vertex number, the sum total of the areas of all the triangles is output on the printer.
3. If you press DEF C , the sum total of all the triangle areas will be cleared and then the program ends.

Note: The number of vertexes may be stored up to the following limits:
CE-212M (or CE-201M) · CE-2H16M (or CE-202M) 255 vertexes

■ REMARKS

$$S=\sqrt{s(s-a)(s-b)(s-c)}$$

$$s=\frac{a+b+c}{2}$$

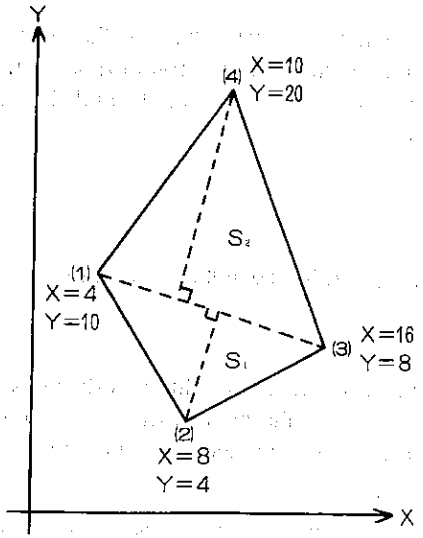


Area of triangle $S=\frac{1}{2}hL$

where
L: Base length (the longest of sides a, b, and c)
h: Height (Figures are truncated to three decimal places.)

■ EXAMPLE

Figure out the area of a 4-sided polygon as shown below.



■ PRINTOUT

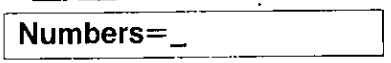
```
Point 1x= 4
Point 1y= 10
Point 2x= 8
Point 2y= 4
Point 3x= 16
Point 3y= 8
Point 4x= 10
Point 4y= 20
  1-2-3
A=            7.211
B=            8.944
C=           12.166
H=            5.260
S=           31.996580
  1-3-4
A=           12.166
B=           13.416
C=           11.662
H=            9.838
S=           65.993304

*TS*=        97.989884
```

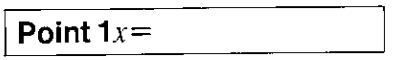
■ KEY OPERATION SEQUENCE

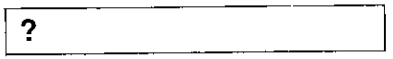
<Input coordinates X and Y of each vertex>

1. DEF A



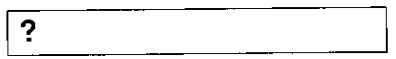
2. 4 ENTER





3. 4 ENTER





⋮

Input data in the same manner as above.

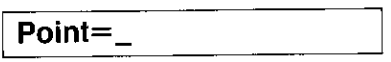
⋮

4. 20 ENTER



<Input vertex (Point) Nos. of each triangle>

1. DEF B



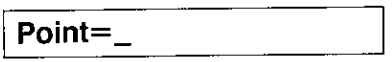
2. 1 ENTER



3. 2 ENTER



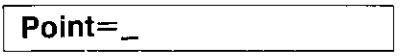
4. 3 ENTER



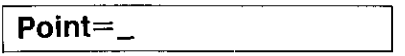
5. 1 ENTER



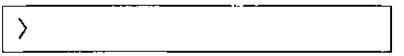
6. 3 ENTER



7. 4 ENTER



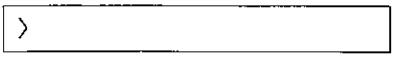
8. ENTER



<Clearing of sum total of triangle areas>

1. DEF C





■ PROGRAM LIST

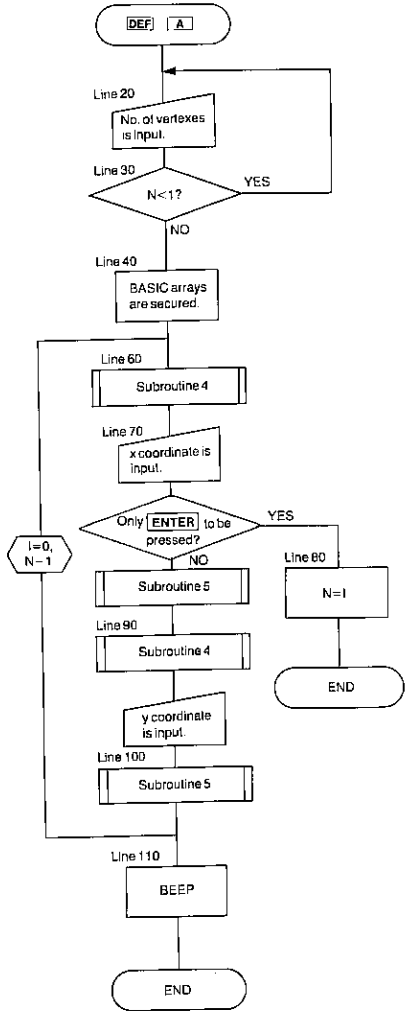
```
10:"A":USING :CLEAR :
   WAIT 0
20:INPUT "Numbers=";N
30:IF N<1 BEEP 2:GOTO 2
   0
40:DIM X(N-1),Y(N-1),B$
   (0)
50:FOR I=0 TO N-1
60:B$(0)="x=":GOSUB 360
70:INPUT X(I):B$(0)="x=
   "+STR$ X(I):GOSUB 3
   70:GOTO 90
80:N=I:END
90:B$(0)="y=":GOSUB 360
   :INPUT Y(I)
100:B$(0)="y= "+STR$ Y(I
   ):GOSUB 370:NEXT I
110:BEEP 1:END
120:"B":USING :INPUT "Po
   int=";O,"Point=";P,"
   Point=";Q:GOTO 140
130:GOTO 310
140:IF (O<1)+(O>N)+(P<1)
   +(P>N)+(Q<1)+(Q>N)<>
   0 GOTO 120
150:C=X(O-1):D=Y(O-1):E=
   X(P-1):F=Y(P-1):G=X(
   Q-1):H=Y(Q-1)
160:X=E-C:Y=F-D:GOSUB 33
   0
170:A=X:X=G-E:Y=H-F:
   GOSUB 330
180:B=X:X=C-G:Y=D-H:
   GOSUB 330
190:C=X:IF A>X LET X=A
200:IF B>X LET X=B
210:I=(A+B+C)/2:S=SQR (I
   *(I-A)*(I-B)*(I-C))
220:J= INT ((2*S/X)*10^3
   )/10^3:L=X:GOSUB 340
230:X=L:S=X*J/2:K=K+S:L=
   A:GOSUB 340
240:A=L:L=B:GOSUB 340
250:B=L:L=C:GOSUB 340
260:C=L:L=S:GOSUB 350
270:S=L:L=K:GOSUB 350
280:K=L:LPRINT "  ";STR$
   O+"-"+STR$ P+"-"+
   STR$ Q
290:LPRINT "A= "; USING
   "##############.###";A
300:LPRINT "B= ";B:
   LPRINT "C= ";C:
   LPRINT "H= ";J:
   LPRINT USING "######
   ######.######";"S= "
   ;S:GOTO 120
310:LPRINT "":LPRINT "*T
   S*="; USING "#######
   ##.######";K:END
320:"C":K=0:USING :PAUSE
   " ** TS CLEAR **":
   END
330:X=SQR (X*X+Y*Y):
   RETURN
340:L= INT (L*1000+.5)/1
   000:RETURN
350:L= INT (L*1000000)/1
   000000:RETURN
360:PAUSE "Point ";STR$
   (I+1);B$(0):RETURN
370:LPRINT "Point ";STR$
   (I+1);B$(0):RETURN
```

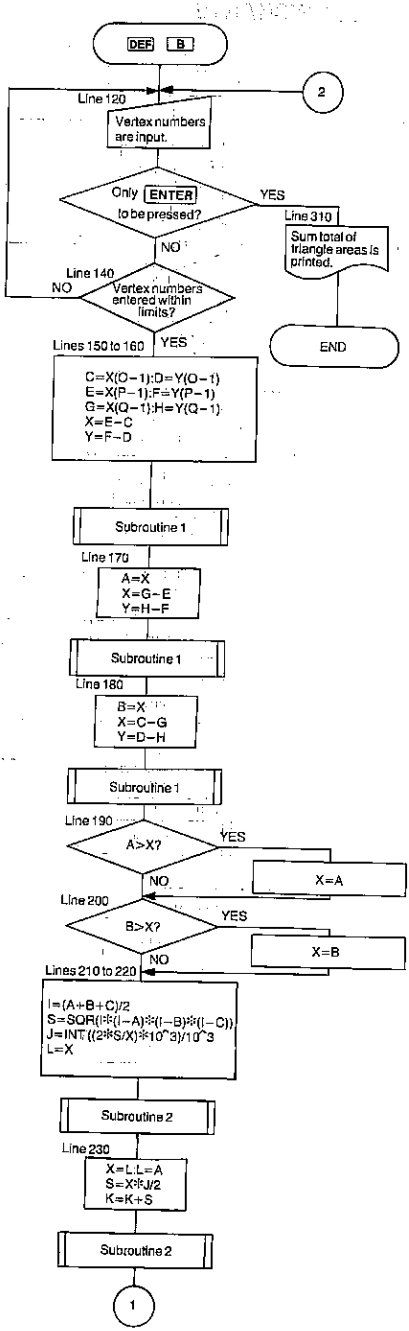
962 Bytes

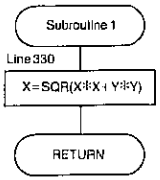
■ MEMORY CONTENTS

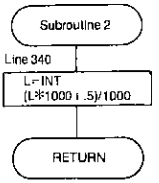
A	a
B	b
C	x_1, √
D	y_1
E	x_2
F	y_2
G	x_3
H	y_3
I	S
J	h
K	Σs
L	√
N	Number of vertexes
O	√
P	√
Q	√
S	S
X	√
Y	√
X (N−1)	x-coordinate
Y (N−1)	y-coordinate
B $(0)	√

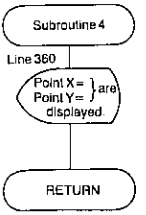
■ FLOWCHARTS

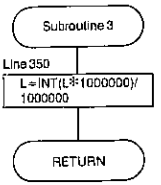


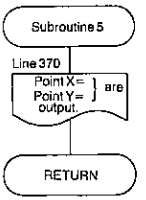

DEF B
2
Line 120
Vertex numbers are input.
Only ENTER to be pressed?
YES
Line 310
Sum total of triangle areas is printed.
END
NO
Line 140
NO
Vertex numbers entered within limits?
YES
Lines 150 to 160
C=X(O-1):D=Y(O-1)
E=X(P-1):F=Y(P-1)
G=X(Q-1):H=Y(Q-1)
X=E-C
Y=F-D
Subroutine 1
Line 170
A=X
X=G-E
Y=H-F
Subroutine 1
Line 180
B=X
X=C-G
Y=D-H
Subroutine 1
Line 190
A>X?
YES
X=A
NO
Line 200
B>X?
YES
X=B
NO
Lines 210 to 220
I=(A+B+C)/2
S=SQR(I*(I-A)*(I-B)*(I-C))
J=INT((2*S/X)*10^3)/10^3
L=X
Subroutine 2
Line 230
X=L:L=A
S=X*J/2
K=K+S
Subroutine 2
1
1
Lines 240 to 250
A=L:L=B
B=L:L=C
Subroutine 2
Lines 260 to 270
C=L:L=S
S=L:L=K
Subroutine 3
Lines 290 to 300
Base length, height, & area of each triangle are printed.
2
DEF C
Line 320
Sum total is cleared. USING is released.
TS CLEAR is displayed.
END


Subroutine 1
Line 330
X=SQR(X*X+Y*Y)
RETURN


Subroutine 2
Line 340
L=INT
(L*1000+.5)/1000
RETURN


Subroutine 4
Line 360
Point X=
Point Y=
are
displayed.
RETURN


Subroutine 3
Line 350
L=INT(L*1000000)/
1000000
RETURN

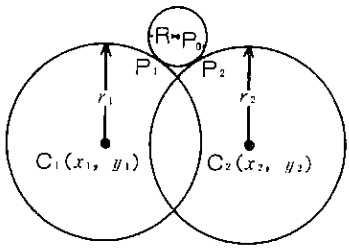

Subroutine 5
Line 370
Point X=
Point Y=
are
output.
RETURN

PROGRAM TITLE: A Circle Osculating Two Circles

There are two adjoining circles to both of which another circle is tenderly adhering. Will a warm feeling begin to bud there? Such a way of looking at these circles may bring a light touch to your study of geometry. This program finds out the center of a circle osculating two circles and the coordinates of the two points of contact by inputting the center coordinates and radius of each of the two circles together with three discriminating conditions.

■ HOW TO OPERATE

1. Press DEF A . (Program starts)
2. Input the center coordinates (x_1 and y_1) of circle C_1 and radius R_1, the center coordinates (x_2 and y_2) of circle C_2 and radius R_2, and the radius R of the circle osculating circle C_1 and C_2 according to the display.

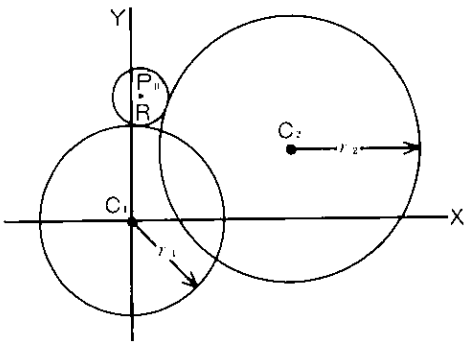


3. Then input the value of each of the following three discriminating conditions:

	Conditions	Value
(1)	When the circle to solve with respect to circle C_1 is osculating externally	1
	When the circle to solve with respect to circle C_1 is osculating internally	−1
(2)	When the circle to solve with respect to circle C_2 is osculating externally	1
	When the circle to solve with respect to circle C_2 is osculating internally	−1
(3)	When the circle is on the left side as circle C_2 is viewed from circle C_1	1
	When the circle is on the right side as circle C_2 is viewed from circle C_1	−1

When all the above data is input, the center coordinates of the circle osculating circles C_1 and C_2 and the coordinates of points of contact P_1 and P_2 are displayed on the screen in the order named. Then, the program ends.

■ EXAMPLE



C1: $x_1=0$, $y_1=0$, $r_1=30$
C2: $x_2=50$, $y_2=20$, $r_2=40$
R=10

Discriminating conditions
(1) 1 (osculating externally)
(2) 1 (osculating externally)
(3) 1 (on left side)

■ KEY OPERATION SEQUENCE

1. DEF A

C1 x=_

2. 0 ENTER

C1 y=_

3. 0 ENTER

C1 r=_

⋮

Input data in the same manner as above.

⋮

4. 10 ENTER

C1 . OUT:1 IN:−1=_

5. 1 ENTER

C2 . OUT : 1 IN:−1=_

6. 1 ENTER

LEFT : 1 RIGHT :1−1=_

7. 1 ENTER

P0 x= 4.08

8. ENTER

P0 y= 39.79

9. ENTER

P1 x= 3.06

10. ENTER

P1 y= 29.84

11. ENTER

P2 x= 13.27

12. ENTER

P2 y= 35.83

13. ENTER

›

■ PROGRAM LIST

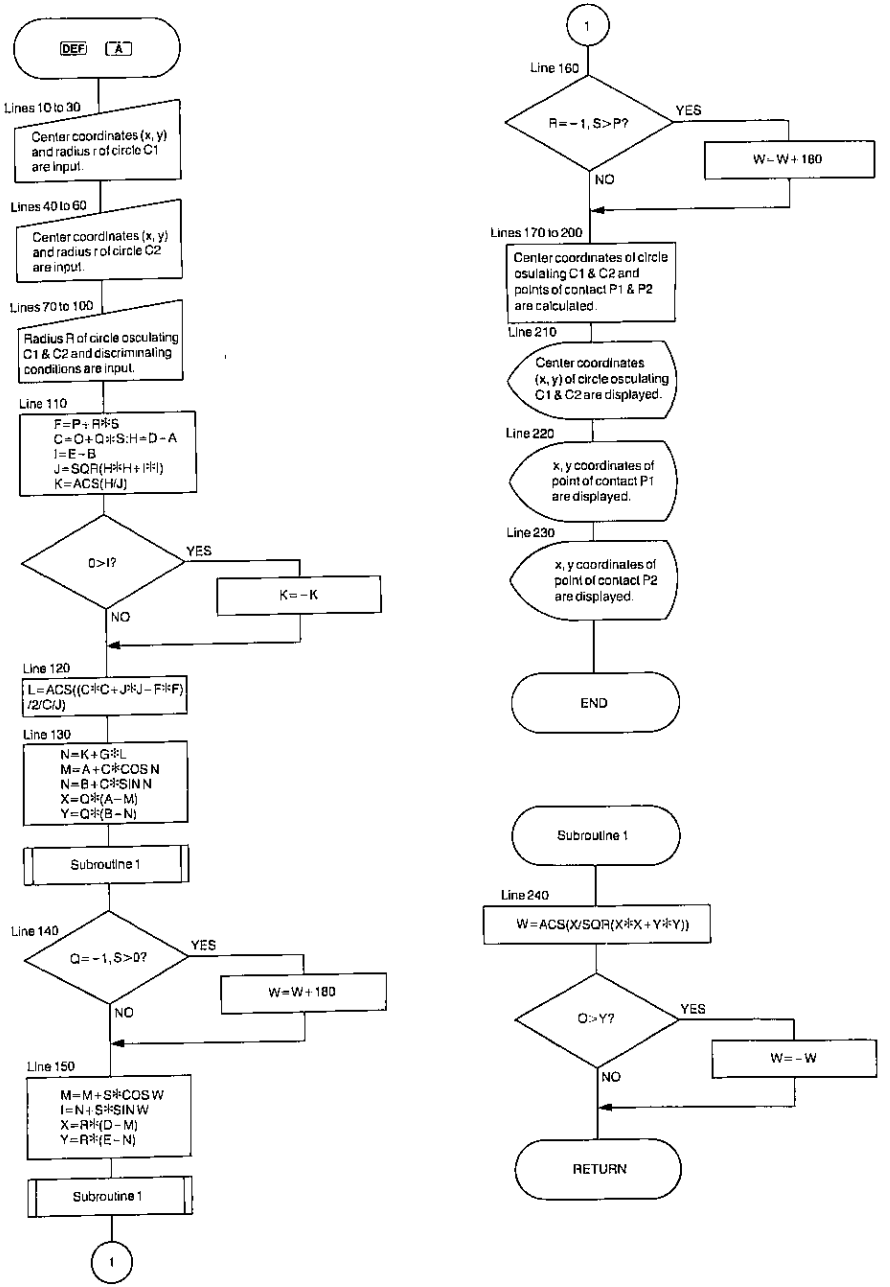
```
 10:"A":DEGREE :INPUT "C
    1 x=";A
 20:INPUT "C1 y=";B
 30:INPUT "C1 r=";O
 40:INPUT "C2 x=";D
 50:INPUT "C2 y=";E
 60:INPUT "C2 r=";P
 70:INPUT "R=";S
 80:INPUT "C1.OUT:1 IN:-
    1=";Q
 90:INPUT "C2.OUT:1 IN:-
    1=";R
100:INPUT "LEFT:1 RIGHT:
    -1=";G
110:F=P+R*S:C=O+Q*S:H=D-
    A:I=E-B:J=SQR (H*H+I
    *I):K=ACS (H/J):IF 0
    >I LET K=-K
120:L=ACS ((C*C+J*J-F*F)
    /2/C/J)
130:N=K+G*L:M=A+C*COS N:
    N=B+C*SIN N:X=Q*(A-M
    ):Y=Q*(B-N):GOSUB 24
    0
140:IF ((Q=-1)*(S>0))=1
    LET W=W+180
150:H=M+S*COS W:I=N+S*
    SIN W:X=R*(D-M):Y=R*
    (E-N):GOSUB 240
160:IF ((R=-1)*(S>P))=1
    LET W=W+180
170:J=M+S*COS W:K=N+S*
    SIN W
180:M=M+SGN M*.005:N=N+
    SGN N*.005
190:H=H+SGN H*.005:I=I+
    SGN I*.005
200:J=J+SGN J*.005:K=K+
    SGN K*.005
210:PRINT "P0 x="; USING
    "########.##";M:
    PRINT "P0 y=";N
220:PRINT "P1 x=";H:
    PRINT "P1 y=";I
230:PRINT "P2 x=";J:
    PRINT "P2 y=";K:END
240:W=ACS (X/SQR (X*X+Y*
    Y)):IF 0>Y LET W=-W
250:RETURN
```

653 Bytes

■ MEMORY CONTENTS

A	x_1
B	y_1
C	√
D	x_2
E	y_2
F	√
G	Discriminating condition (3)
H	P_1x
I	P_1y
J	P_2x
K	P_2y
L	√
M	P_0x
N	P_0y
O	r_1
P	r_2
Q	Discriminating condition (1)
R	Discriminating condition (2)
S	R
W	√
X	√
Y	√

■ FLOWCHART


DEF A
Lines 10 to 30
Center coordinates (x, y) and radius r of circle C1 are input.
Lines 40 to 60
Center coordinates (x, y) and radius r of circle C2 are input.
Lines 70 to 100
Radius R of circle osculating C1 & C2 and discriminating conditions are input.
Line 110
F=P÷R*S
C=O+Q*S:H=D-A
I=E-B
J=SQR(H*H+I*I)
K=ACS(H/J)
0>I?
YES
K=-K
NO
Line 120
L=ACS((C*C+J*J-F*F)/2/C/J)
Line 130
N=K+G*L
M=A+C*COS N
N=B+C*SIN N
X=Q*(A-M)
Y=Q*(B-N)
Subroutine 1
Line 140
Q=-1, S>0?
YES
W=W+180
NO
Line 150
M=M+S*COS W
I=N+S*SIN W
X=R*(D-M)
Y=R*(E-N)
Subroutine 1
1
1
Line 160
R=-1, S>P?
YES
W=W+180
NO
Lines 170 to 200
Center coordinates of circle osculating C1 & C2 and points of contact P1 & P2 are calculated.
Line 210
Center coordinates (x, y) of circle osculating C1 & C2 are displayed.
Line 220
x, y coordinates of point of contact P1 are displayed.
Line 230
x, y coordinates of point of contact P2 are displayed.
END
Subroutine 1
Line 240
W=ACS(X/SQR(X*X+Y*Y))
O>Y?
YES
W=-W
NO
RETURN

PROGRAM TITLE: Frequency Characteristics Graph

Required Peripheral Equipment
CE-140P or CE-515P
(or CE-516P) and CE-516L

The Frequency Characteristics Graph is well known among audio equipment maniacs. By giving a transmission function to the computer, this program presents the frequency characteristics of an audio equipment component in a graphic form. A clear, easy-to-see color graph is output on the printer, utilizing the very feature of the computer.

■ HOW TO OPERATE

1. Press DEF A . (Program starts)
   The axes of the graph are output on the printer.
2. Input the pitch value (step) and maximum value of gain and the minimum value of frequency according to the display. The scale divisions of the graph are output on the printer.
3. Then, input the values of the frequency step, degree of the transmission function, and the respective coefficients for the denominator and numerator of the formula (see REMARKS below). The frequency characteristics graph is output on the printer.
4. Next, determine whether you need to output the coefficient data of the formula on the printer or calculate another transmission function.
   - When the coefficient data is to be printed out .. Press Y ENTER .
     (In this case, the program ends after the coefficient data has been printed.)
   - When another transmission function is to be calculated ..Press N ENTER .
     (In this case, input the data in the same manner as in step 3 above and the graph utilizing another transmission function will be printed out. Determine again whether you need to output the data of the function or calculate the transmission function.)

■ REMARKS

Formula:

$$G(S)=\frac{B(0)+B(1)S+\cdots\cdots+B(N-1)S^{N-1}}{A(0)+A(1)S+A(2)S^2+\cdots\cdots+A(N)S^N} \quad (N\leq 12)$$

where the frequency is f(Hz), the following formula may be derived from the above.

$$G(j2\pi f)=\frac{B(0)+B(1)\cdot j2\pi f+\cdots\cdots+B(N-1)\cdot(j2\pi f)^{N-1}}{A(0)+A(1)\cdot j2\pi f+\cdots\cdots+A(N(j2\pi f)^N)}$$

$$=\frac{B(0)-B(2)\cdot(2\pi f)^2+B(4)\cdot(2\pi f)^4+\cdots\cdots+j\{B(1)\cdot 2\pi f-B(3)\cdot(2\pi f)^3+\cdots\cdots\}}{A(0)-A(2)\cdot(2\pi f)^2+\cdots\cdots+j\{A(1)\cdot 2\pi f-A(3)\cdot(2\pi f)^3+\cdots\cdots\}}$$

Then, the formula for gain G(dB) may be given as follows:

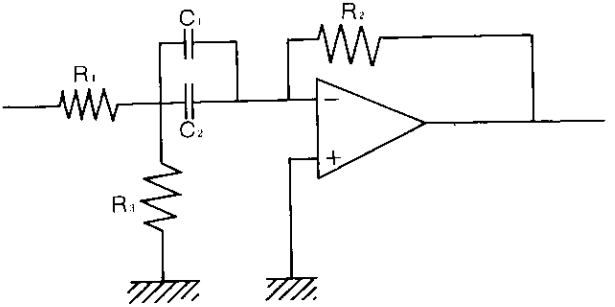
$$G=20\log|G(j2\pi f)|$$

$$G=20\log\sqrt{\frac{|B(0)-B(2)\cdot(2\pi f)^2+\cdots|^2+|B(1)\cdot 2\pi f-B(3)\cdot(2\pi f)^3+\cdots|^2}{|A(0)-A(2)\cdot(2\pi f)^2+\cdots|^2+|A(1)\cdot 2\pi f-A(3)\cdot(2\pi f)^3+\cdots|^2}}$$

$$=10\log\frac{|B(0)-B(2)\cdot(2\pi f)^2+\cdots|^2+|B(1)\cdot 2\pi f-B(3)\cdot(2\pi f)^3+\cdots|^2}{|A(0)-A(2)\cdot(2\pi f)^2+\cdots|^2+|A(1)\cdot 2\pi f-A(3)\cdot(2\pi f)^3+\cdots|^2}$$

■ EXAMPLE

Draw a frequency characteristics graph for a filter (bandpass filter) configured as shown below.



$$G(S)=-\frac{\frac{S}{R_1C_1}}{S^2+\frac{C_1+C_2}{R_3C_1C_2}S+\frac{1}{R'R_3C_1C_2}} \qquad R'\leqq\frac{R_1R_2}{R_1+R_2}$$

Pitch value of gain: 25[dB]
Max. value of gain: 50[dB]

Min. value of frequency: 1[Hz]

where $R_1=2k\Omega$、$R_2=660\Omega$、$R_3=100k\Omega$
$C_1=0.01\mu F$、$C_2=0.1\mu F$

Frequency step = 80
Degree of transmission function = 2
Coefficients for denominator/numerator

$A(0)=2015152$ $B(0)=0$
$A(1)=1100$ $B(1)=50000$
$A(2)=1$

where $R_1=2k\Omega$、$R_2=660\Omega$、$R_3=200k\Omega$
$C_1=C_2=0.1\mu F$

Frequency step = 120
Degree of transmission function = 2
Coefficients for denominator/numerator

A(0)=1007576 B(0)=0
A(1)=100 B(1)=5000
A(2)=1

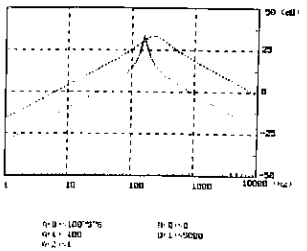
■ When using CE-515P (or CE-516P)

Modify line 10 of the program list on page 318 as follows:

```
10:"A": CLOSE : OPEN "L200,N,8,1,A,L":
   CONSOLE 160: USING :C=0: CLEAR
```

■ PRINTOUT

(Printed out in color.)



■ KEY OPERATION SEQUENCE

1. DEF A

```
GAIN STEP (dB)_
```

2. 25 ENTER

```
GAIN MAX. (dB)_
```

3. 50 ENTER

```
FREQ. MIN. (Hz)_
```

4. 1 ENTER

```
FREQ . STEP=
?
```

5. 80 ENTER

```
DEGREE =
?
```

6. 2 ENTER

```
A (Ø) =
?
```

7. 2015152 ENTER

```
A (1) =
?
```

⋮

Input data in the same manner as above.

⋮

8. 50000 ENTER

```
PRINTOUT Y/N_
```

9. N ENTER

```
FREQ. STEP=
?
```

10. 120 ENTER

```
DEGREE =
?
```

11. 2 ENTER

```
A (Ø) =
?
```

12. 1007576 ENTER

```
A (1) =
?
```

13. 100 ENTER

```
A (2) =
?
```

⋮

Input data in the same manner as above.

⋮

14. 5000 ENTER

```
PRINTOUT Y/N_
```

15. Y ENTER

```
>
```

■ PROGRAM LIST

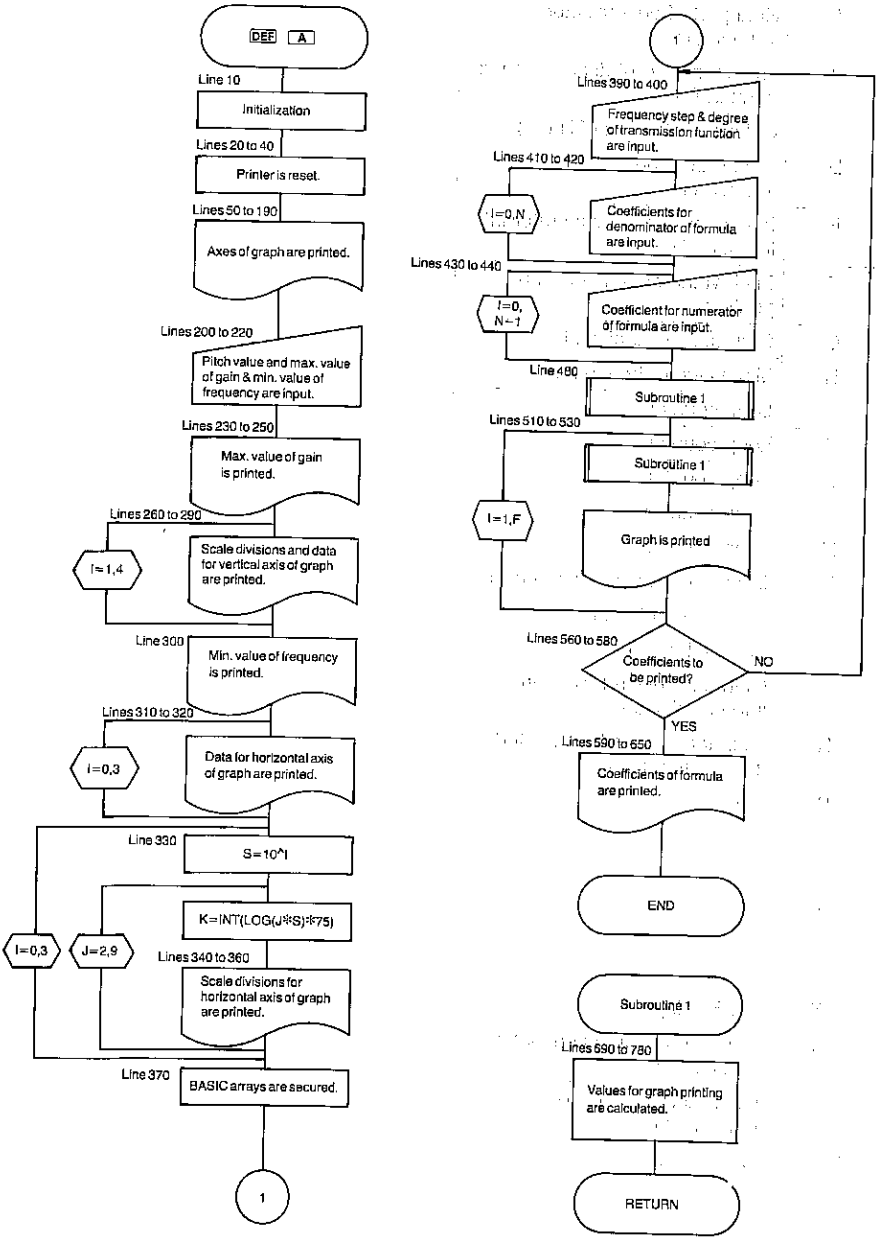
```
10: A$: CONSOLE 160: USING :C=0: CLEAR
20:GTEXT : LPRINT
30:LF 2
40:GRAPH : COLOR 0
50:GLCURSOR (90,0)
60:SORGN
70:LLINE -(300,0)-(300,-200)-(0,-200)-
   (0,0),0
80:GLCURSOR (0,-50)
90:LLINE -(300,-50),3
100:GLCURSOR (300,-100)
110:LLINE -(0,-100),3
120:GLCURSOR (0,-150)
130:LLINE -(300,-150),3
140:GLCURSOR (225,-200)
150:LLINE -(225,0),3
160:GLCURSOR (150,0)
170:LLINE -(150,-200),3
180:GLCURSOR (75,-200)
190:LLINE -(75,0),3
200:INPUT "GAIN STEP (dB)";D
210:INPUT "GAIN MAX. (dB)";P
220:INPUT "FREQ. MIN.(Hz)";M
230:CROTATE 0
240:CSIZE 1
250:GLCURSOR (308,-5): LPRINT "P";STR$
   P;" (dB)"
260:FOR I=1 TO 4
270:GLCURSOR (297,-(25+(I-1)*50))
280:LLINE -(300,-(25+(I-1)*50))
290:GLCURSOR (305,-I*50): LPRINT "P";
   STR$ (P-I*D): NEXT I
300:GLCURSOR (285,-210): LPRINT "P";
   STR$ (M*10^4);" (Hz)"
310:FOR I=0 TO 3
320:GLCURSOR (I*75,-215): LPRINT "P";
   STR$ (M*10^I): NEXT I
330:FOR I=0 TO 3:S=10^I: FOR J=2 TO 9:K
   = INT (LOG (J*S)*75)
340:GLCURSOR (K,-196)
350:LLINE -(K,-200)
360:NEXT J: NEXT I
370:DIM A(15),B(15),W(15)
380:C=3:N=1
390:PAUSE "FREQ. STEP=": INPUT F
400:PAUSE "DEGREE=": INPUT N
410:FOR I=0 TO N: PAUSE "A("+STR$ I+")=
   ": INPUT A(I)
420:NEXT I
430:FOR I=0 TO N-1: PAUSE "B("+STR$ I+"
   )=": INPUT B(I)
440:NEXT I
450:FOR I=N+1 TO 15:A(I)=0:B(I)=0: NEXT
   I
460:B(N)=0
470:I=0
480:R=D*4: GOSUB 690
490:GLCURSOR (0,-G):I=300/F
500:COLOR C
510:FOR I=1 TO F: GOSUB 690
520:LLINE -(I*INT(I),-G),0
530:NEXT I
540:GLCURSOR (0,-250)
550:C=C-1: IF C>0 THEN 590
560:Z$="": INPUT "PRINTOUT Y/N";Z$
570:IF Z$="Y" THEN GOTO 590
580:GOTO 390
590:GTEXT : LPRINT
600:CSIZE 1
610:FOR N=0 TO N-1: LPRINT USING "&&&&&
   &&&&&&&&&&&&&&&&&";"";
620:LPRINT "A("+STR$ N+")="+STR$ A(N);
630:LPRINT "B("+STR$ N+")="+STR$ B(N)
640:NEXT N
650:LPRINT " ";"A("+STR$ N+")="+STR$ A(
   N)
660:USING
670:CSIZE 2: COLOR 0
680:END
690:S=M*10^(I*(F/4))*2*PI
700:W(0)=1: FOR N=1 TO N:W(N)=W(N-1)*S:
   NEXT N
710:A=0:B=0:D=0:E=0
720:FOR L=0 TO N STEP 4:U=W(L):V=W(L+1)
   :W=W(L+2):X=W(L+3)
730:A=A+A(L)*U-A(L+2)*W
740:B=B+A(L+1)*U-A(L+3)*X
750:D=D+B(L)*U-B(L+2)*W
760:E=E+B(L+1)*U-B(L+3)*X
770:NEXT L
780:G= INT ((P-10*LOG ((D*D+E*E)/(A*A+B
   *B)))*200/R): RETURN
```

1456 Bytes

■ MEMORY CONTENTS

A	Real part of denominator in formula (1)
B	Imaginary part of denominator in formula (1)
C	Color specification of line(s)
D	Real part of numerator in formula (1)
E	Imaginary part of numerator in formula (1)
F	Pitch value of data for graph printing
G	Gain (dB) for a given f
H	Min. value of frequency
I	Loop counter
J	Loop counter
K	Loop counter
L	Loop counter
M	Loop counter
N	Degree of transmission function
O	Pitch value of gain
P	Max. value of gain
Q	Gain for a given f (converted into coordinates)
R	Gain for a given f (converted into coordinates)
S	$2\pi f$
T	V
U	V
V	V
W	V
X	V
Z$	For key input
A (15)	Coefficients for denominator of formula (1)
B (15)	Coefficients for numerator of formula (1)
W (15)	$(W_0)^t=(2\pi f)^t$

■ FLOWCHART


DEF A
Line 10
Initialization
Lines 20 to 40
Printer is reset.
Lines 50 to 190
Axes of graph are printed.
Lines 200 to 220
Pitch value and max. value of gain & min. value of frequency are input.
Lines 230 to 250
Max. value of gain is printed.
Lines 260 to 290
I=1,4
Scale divisions and data for vertical axis of graph are printed.
Line 300
Min. value of frequency is printed.
Lines 310 to 320
I=0,3
Data for horizontal axis of graph are printed.
Line 330
S=10^I
K=INT(LOG(J*S)*K75)
I=0,3
J=2,9
Lines 340 to 360
Scale divisions for horizontal axis of graph are printed.
Line 370
BASIC arrays are secured.
1
1
Lines 390 to 400
Frequency step & degree of transmission function are input.
Lines 410 to 420
I=0,N
Coefficients for denominator of formula are input.
Lines 430 to 440
I=0, N=-1
Coefficient for numerator of formula are input.
Line 480
Subroutine 1
Lines 510 to 530
Subroutine 1
I=1,F
Graph is printed
Lines 560 to 580
Coefficients to be printed?
NO
YES
Lines 590 to 650
Coefficients of formula are printed.
END
Subroutine 1
Lines 690 to 780
Values for graph printing are calculated.
RETURN

PROGRAM TITLE: Circular Graph

Required Peripheral Equipment
CE-140P or CE-515P
(or CE-516P) and CE-516L

The Circular Graph is a graph that is very useful in figuring out the composition ratio of individual elements to the entire elements. For example, this graph is convenient for looking into the percentage of each budget quota in relation to the total budget or the percentage share a certain product occupies to its total market. The program introduced here executes percentage (%) calculation upon data input, and at the same time draws a circular graph based on the input data and can even show certain data in cutout (detached) form so that it stands out from the graph. Therefore, with this program, it is easier to confirm percentage or emphasize special data.

■ HOW TO OPERATE

1. Press [DEF] [A] . (Program starts)
2. Input the title of the graph, number of data, item title, and data title according to the display.
3. Then, input the data.
   An item No. will be displayed on the screen. So, input the name of the item corresponding to the item number and the data related to that item. Repeat this process as required for the number of data.
4. Determine whether or not you need to correct the data which has been input.
   - When to be corrected: Press [Y] [ENTER] .
   - When not to be corrected: Press [N] [ENTER] .

   (In this case, you can move to step 5.)

   * Method of data correction
   Input the data No. of the data which you want to correct and then input the correct data. After this process, determine whether or not you need to correct the data again.
5. Determine whether or not all the data entered need to be sorted.
   - When to be sorted: Press [Y] [ENTER] .
   - When not to be sorted: Press [N] [ENTER] .
6. Determine whether or not you need to have a doughnut-shaped circular graph.
   - When to be doughnut-shaped: Press [Y] [ENTER] .
   - When not to be doughnut-shaped: Press [N] [ENTER] .

7. Determine whether or not you need to have a cutout section from the graph.
   - When to be cut out: Press [Y] [ENTER] .
   - When not to be cut out: Press [N] [ENTER] .

   (In this case, you can move to step 8.)

   * How to specify a cutout section.

   Input the data number of the item for which you want to have a cutout section.
8. Determine whether or not all data entered are tabulated.
   - When to be tabulated: Press [Y] [ENTER] .
   - When not to be tabulated: Press [N] [ENTER] .

Upon completing the above step, the table of data and circular graph or only the circular graph are output on the printer. Then, the program ends.

Notes:

- Graph title – 10 characters max.
- Number of Data – 2 to 20 characters
- Item title, data title – 10 characters max.
- Item name – 10 characters max.

■ EXAMPLE

You must represent the total sales at each branch of the BCD Inc., on a doughnut-shaped circular graph. Draw the circular graph, so that only the data of the L.A. branch stands out from the graph.

Branch name	N.Y.	L.A.	Chicago
Total sales amount	630	297	270

Boston	Detroit	Seattle
219	156	98

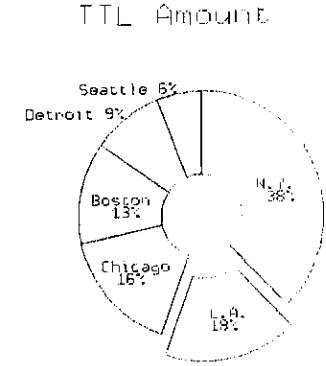
■ When using CE-515P (or CE-516P)

Add line 55 to the program list on page 325 as follows:

```
55:CLOSE : OPEN "1200,N,8,1,A,L"
```

■ **PRINTOUT**

(Printed out in color. Refer to page I.)



Branch	Sales
N.Y.	630
L.A.	297
Chicago	270
Boston	219
Detroit	156
Seattle	98

■ KEY OPERATION SEQUENCE

1. DEF A

```
** CIRCLE GRAPH **
Graph title =_
```

2. TTL SPC
   A SML MOUNT ENTER

```
No. of data=_
```

3. 6 ENTER

```
Item title=_
```

4. B SML RANCH ENTER

```
Data title=_
```

5. S SML ALES ENTER

```
No. 1
Item name=_
```

6. N.Y. ENTER

```
Data=_
```

7. 630 ENTER

```
NO. 2
Item name=_
```

8. L.A. ENTER

```
Data=_
```

9. 297 ENTER

```
No. 3
Item name=_
```

Input data in the same manner as above.

10. 98 ENTER

```
Correct? (Y/N)_
```

11. Y ENTER

```
Data No.=_
```

12. 3 ENTER

```
Data=_
```

13. 270 ENTER

```
Correct? (Y/N)_
```

14. N ENTER

```
Data sort (Y/N)_
```

15. N ENTER

```
Doughnut (Y/N)_
```

16. Y ENTER

```
Detaching (Y/N)_
```

17 Y ENTER

```
Detaching No.=_
```

18. 2 ENTER

```
Draw tbl. (Y/N)_
```

19. Y ENTER

```
>
```

■ PROGRAM LIST

```
10:"A": CONSOLE 80: USING
20:PAUSE "**CIRCLE GRAPH**"
30:CLEAR : RADIAN
40:GOSUB 970
50:P0=PI /2:P1=0:P2=3*PI /2:P3=PI
60:LTEXT : LPRINT
70:COLOR 0
80:GRAPH
90:MS=LEN TT$:MJ=24
100:IF MS*MJ<2X THEN 140
110:MJ=18: IF MS*MJ<2X THEN 140
120:MJ=12: IF MS*MJ<2X THEN 140
130:MJ=6
140:GLCURSOR (PX-RR,-30)
150:IF MJ=6 THEN CSIZE 1: GOTO 190
160:IF MJ=12 THEN CSIZE 2: GOTO 190
170:IF MJ=18 THEN CSIZE 3: GOTO 190
180:CSIZE 4
190:LPRINT "P";TT$
200:CSIZE 2
210:GLCURSOR (PX,PY-30)
220:SORGN
230:FOR K=1 TO YK
240:C(K)= INT (DD(K)/YT*10000+.5)*100
250:NEXT K
260:PP=PI *2/YT:P(0)=0
270:FOR K=1 TO YK
280:P(K)=P(K-1)+PP*DD(K)
290:NEXT K
300:I1= INT (RD*COS P0):I2= INT (RD*SIN
    P0)
310:O1= INT (RR*COS P0):O2= INT (RR*SIN
    P0)
320:YO=99999:Q=0
330:FOR J=1 TO YK
340:IF Q=4 LET Q=0
350:T1=P0-P(J-1): IF T1<0 LET T1=T1+2*
    PI
360:T2=P0-P(J): IF T2<0 LET T2=T2+2*PI
370:TM=P0-P(J-1)-(P(J)-P(J-1))/2
380:IF TM<0 LET TM=TM+2*PI
390:AX=0:AY=0: IF J<>NN THEN 420
400:T=P0-P(J-1)-(P(J)-P(J-1))/2
410:AX= INT (25*COS T):AY= INT (25*SIN
    T):DX=AX:DY=AY
420:T=P0-P(J): COLOR Q
430:O3= INT (RR*COS T):O4= INT (RR*SIN
    T)
440:I3= INT (RD*COS T):I4= INT (RD*SIN
    T)
450:IF J=1 OR J=NN OR J-1=NN THEN 470
460:GOTO 490
470:LLINE (I1+AX,I2+AY)-(O1+AX,O2+AY),0
480:GOTO 500
490:GLCURSOR (O1+AX+1,O2+AY+1)
500:T=P(J-1): COLOR Q
510:LLINE -(RR*COS (P0-T)+AX,RR*SIN (P0
    -T)+AY)
520:T=T+.05: IF T<=P(J) THEN 510
530:LLINE -(O3+AX,O4+AY)
540:IF J=YK AND J=NN THEN 560
550:IF J=YK THEN 570
560:LLINE -(I3+AX,I4+AY)
570:IF RD=0 THEN 620
580:T=P(J): COLOR Q
590:LLINE -(RD*COS (P0-T)+AX,RD*SIN (P0
    -T)+AY)
600:T=T-.05: IF T>=P(J-1) THEN 590
610:LLINE -(I1+AX,I2+AY)
620:MJ=12: IF RR<100 LET MJ=6
630:IF MJ=6 THEN CSIZE 1
640:IF MJ=12 THEN CSIZE 2
650:A= INT (C(J)/10000+.5)
660:M=LEN Y$(J):A$=STR$ A:N=LEN A$+2
670:HH=(M+N)*MJ: COLOR 0
680:GOSUB 1630: IF SW=1 THEN 910
690:XM= INT (RR*COS TM)+AX:YM= INT (RR*
    SIN TM)+AY
700:IF ABS (YO-YM)<MJ+2 THEN 800
710:XL=0:YL=0
720:IF TM>P0 THEN 740
730:XS=XM:YS=YM: GOTO 790
740:IF TM>P3 THEN 760
750:XS=XM-HH:YS=YM: GOTO 790
760:IF TM>P2 THEN 780
770:XS=XM-HH:YS=YM-MJ: GOTO 790
780:XS=XM:YS=YM-MJ
790:YO=YS: GOTO 880
800:IF TM>P0 THEN 820
810:XS=XM:YS=YO-MJ-2:XL=XS:YL=YS+MJ:
    GOTO 870
820:IF TM>P3 THEN 840
830:XS=XM:YS=YO+MJ+2:XL=XS:YL=YS: GOTO
    870
840:IF TM>P2 LET YS=YO-MJ-2: GOTO 860
850:YS=YO+MJ+2
860:XS=XM:XL=XS:YL=YS+MJ
870:YO=YS: COLOR 0
880:GLCURSOR (XS,YS): LPRINT "P";Y$(J);
    " ";A$;"%"
890:IF XL=0 AND YL=0 THEN 910
900:LLINE (XM,YM)-(XL,YL)
910:O1=O3:O2=O4:I1=I3:I2=I4
920:Q=Q+1: NEXT J: DEGREE
930:IF F$="Y" GOSUB 1510: GOTO 950
940:GLCURSOR (-PX,-RR-100)
950:LTEXT : LPRINT
960:LF 5: END
970:INPUT "Graph title=";TT$: IF LEN TT
    $>10 OR TT$="" THEN 970
980:INPUT "No. of data=";YK: IF YK<2 OR
    YK>20 THEN 980
990:DIM DD(YK),Y$(YK),X$(2),P(YK),C(YK)
 1000:INPUT "Item title=";X$(1): IF LEN
      X$(1)>10 OR X$(1)="" THEN 1000
 1010:INPUT "Data title=";X$(2): IF LEN
      X$(2)>10 OR X$(2)="" THEN 1010
 1020:FOR L=1 TO YK
 1030:PAUSE "No.";STR$ L
 1040:INPUT "Item name=";Y$(L)
 1050:IF LEN Y$(L)>10 OR Y$(L)="" THEN
      1030
 1060:INPUT "Data=";DD(L): IF DD(L)=0
      THEN 1060
 1070:NEXT L
 1080:INPUT "Correct? (Y/N)";A$
```

```
1090:IF A$="N" THEN 1150
1100:IF A$<>"Y" THEN 1080
1110:INPUT "Data No.=";L
1120:IF L<1 OR L>YK THEN 1110
1130:INPUT "Data=";DD(L): IF DD(L)=0
     THEN 1130
1140:GOTO 1080
1150:INPUT "Data sort (Y/N)";A$
1160:IF A$="N" THEN 1320
1170:IF A$<>"Y" THEN 1150
1180:I=1
1190:K=I+1
1200:IF I=YK AND K>=YK THEN 1320
1210:IF DD(I)>DD(K) THEN 1240
1220:W=DD(I):DD(I)=DD(K):DD(K)=W
1230:A$=Y$(I):Y$(I)=Y$(K):Y$(K)=A$
1240:K=K+1
1250:IF I=YK AND K>YK THEN 1320
1260:IF K>YK THEN 1280
1270:GOTO 1210
1280:I=I+1
1290:IF I=YK AND K>YK THEN 1320
1300:IF I>YK THEN 1320
1310:GOTO 1190
1320:YT=0
1330:FOR L=1 TO YK:YT=YT+DD(L): NEXT L
1340:WX=480:WY=2500
1350:C$="B"
1360:ZX=300:RR=150:AH=12/7:ZY=ZX
1370:INPUT "Doughnut (Y/N)";E$
1380:IF E$="N" LET RD=0:NO=0: GOTO 141
     0
1390:IF E$<>"Y" THEN 1370
1400:RD= INT (20/2.5)
1410:INPUT "Detaching (Y/N)";A$
1420:IF A$="N" THEN 1460
1430:IF A$<>"Y" THEN 1410
1440:INPUT "Detaching No.=";NN
1450:IF NN<1 OR NN>YK THEN 1440
1460:INPUT "Draw tbl. (Y/N)";F$
1470:IF F$="Y" OR F$="N" THEN 1490
1480:GOTO 1460
1490:PY=- INT (80+ZY/2):PX= INT (WX/2)
1500:RETURN
1510:Y=-RR-50
1520:GLCURSOR (0,Y)
1530:LTEXT : LPRINT
1540:LPRINT USING "&&&&&&&&";X$(L);
1550:LPRINT USING "&&";" ";
1560:LPRINT USING "&&&&&&&&";X$(2)
1570:FOR k=1 TO Yk
1580:LPRINT USING "&&&&&&&&";Y$(k);
1590:LPRINT USING "&&";" ";
1600:LPRINT USING "#########";DD(k)
1610:NEXT k
1620:RETURN
1630:R=(RR+RD)/2:XM= INT (R*COS TM):YM
     = INT (R*SIN TM)
1640:X1=YM/TAN T1:X2=YM/TAN T2:XC=SQR
     (RR*RR-YM*YM)
1650:IF T1>P0 THEN 1700
1660:IF T2<P0 LET PT=1: GOTO 1780
1670:IF T2<P3 LET PT=4: GOTO 1780
1680:IF T2<P2 LET PT=3: GOTO 1780
1690:PT=2: GOTO 1780
1700:IF T1<P2 THEN 1740
1710:IF T2>P2 LET PT=5: GOTO 1780
1720:IF T2>P3 LET PT=6: GOTO 1780
1730:PT=7: GOTO 1780
1740:IF T1<P3 THEN 1770
1750:IF T2>P3 LET PT=8: GOTO 1780
1760:PT=9: GOTO 1780
1770:PT=10
1780:IF PT=2 OR PT=3 OR PT=4 OR PT=5
     OR PT=9 THEN 1800
1790:GOTO 1850
1800:GLCURSOR (XM-(M+MJ)/2-AX,YM-MJ/2+
     AY)
1810:LPRINT "P";Y$(J)
1820:GLCURSOR (XM-MJ+AX,YM-MJ/2+AY-MJ-
     3)
1830:LPRINT "P";A$;"%"
1840:SW=1: RETURN
1850:IF PT=1 LET XS=X1:XE=X2: IF X2>XC
     LET XE=XC
1860:IF PT=5 LET XS=X2:XE=X1: IF X1>XC
     LET XE=XC
1870:IF PT=6 LET XS=X2:XE=X1: IF X2<-X
     C LET XS=-XC
1880:IF PT=6 AND X1>XC LET XE=XC
1890:IF PT=8 LET XS=X2:XE=X1: IF X2<-X
     C LET XS=-XC
1900:IF PT=10 LET XS=X1:XE=X2: IF X1<-
     XC LET XS=-XC
1910:XH=XE-XS-2*MJ: IF XH<M*MJ LET SW=
     0: RETURN
1920:GLCURSOR (XS+MJ+(XH-M*MJ)/2+AX,YM
     +AY)
1930:LPRINT "P";Y$(J)
1940:GLCURSOR (XS+XH/2-MJ+AX,YM-MJ-3+A
     Y)
1950:LPRINT "P";A$;"%"
1960:SW=1: RETURN
```

3926 Bytes

■ MEMORY CONTENTS

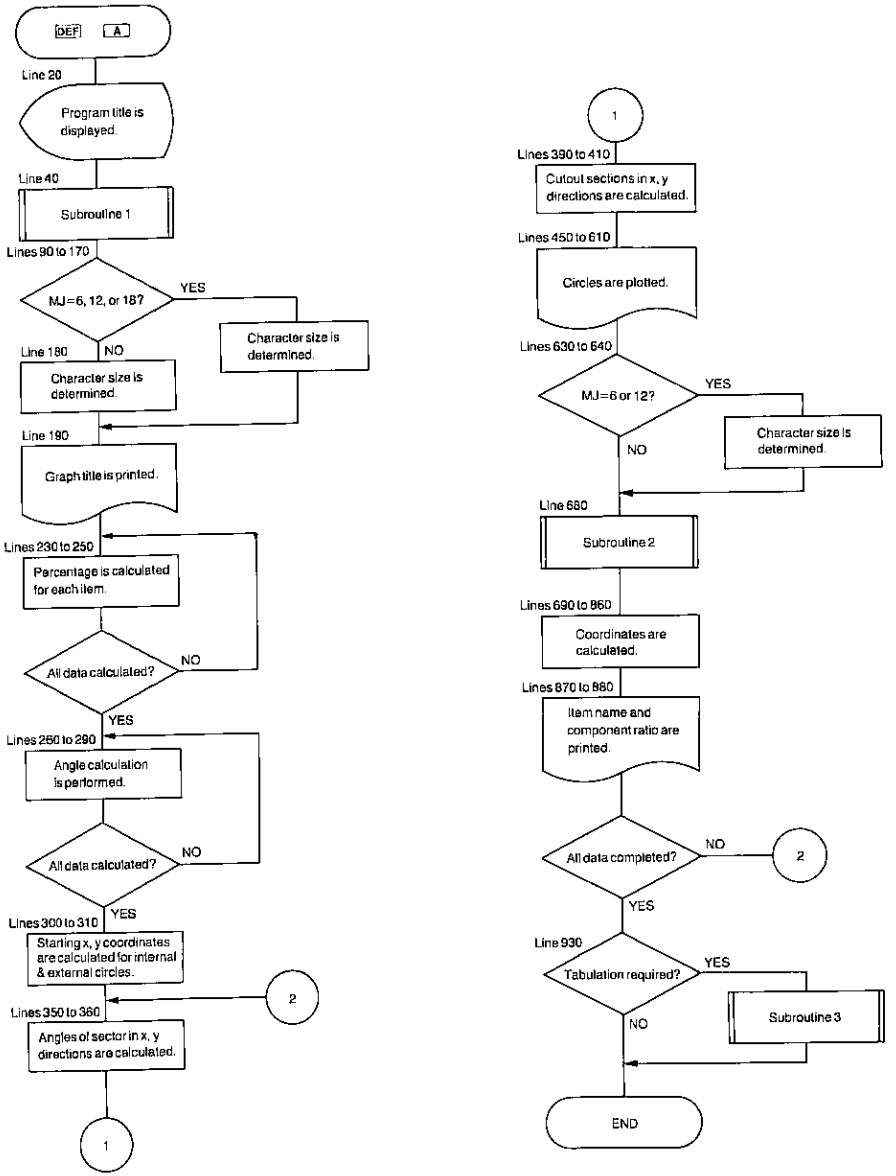
<variables (numeric)>

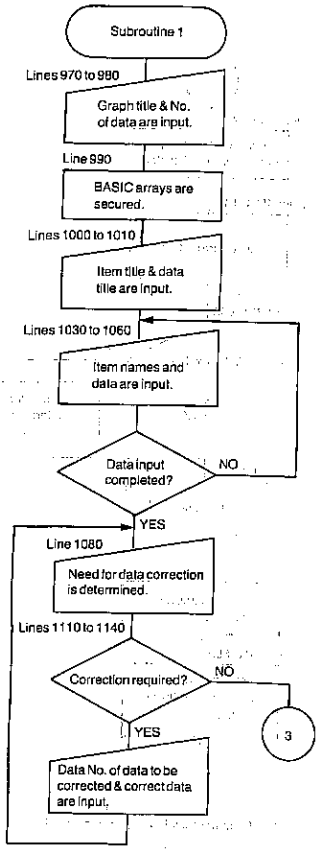
A	√
I	√
J	Loop counter
K	√, loop counter
L	Loop counter, √
M	√
N	√
Q	√
R	√
T	√
W	√
Y	√, print position of table
C(YK)	Each data for which percentage has been calculated
DD(YK)	Each data
P(YK)	Angle calculation value of each data
AH	√
AX	Cutout in x-direction
AY	Cutout in y-direction
DX	Cutout in x-direction
DY	Cutout in y-direction
HH	√
I1	Starting x-coordinate of internal circle
I2	Starting y-coordinate of internal circle
I3	√
I4	√
MJ	Character size of graph and item titles
MS	Number of characters for graph title
NN	Detaching No.
NO	√
O1	Starting x-coordinate of external circle
O2	Starting y-coordinate of external circle
O3	√
O4	√
P0	$\pi/2$
P1	0
P2	$3*\pi/2$
P3	π
PP	Angle calculation
PT	√
PX	Base position of graph in x-direction
PY	Base position of graph in y-direction
RD	Diameter of internal circle
RR	Diameter of external circle
SW	√
T1	Angle of sector in x-direction
T2	Angle of sector in y-direction
TM	√
WX	Paper size in x-direction
WY	Paper size in y-direction
X1	√
X2	√
XC	√
XE	√
XL	√
XM	√
XS	√
YK	Number of data
YL	√
YM	√
YO	√
YS	√
YT	Data summary
ZX	√
ZY	√

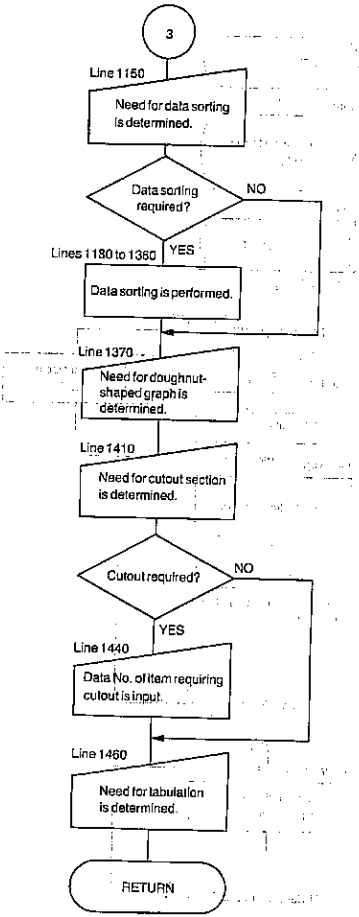
<variables (string)>

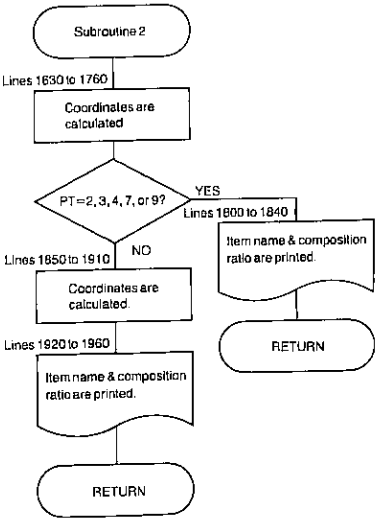
A$	√
C$	√
F$	√
X$(1)	Item title
X$(2)	Data title
Y$(YK)	Item name
TT$	Graph title

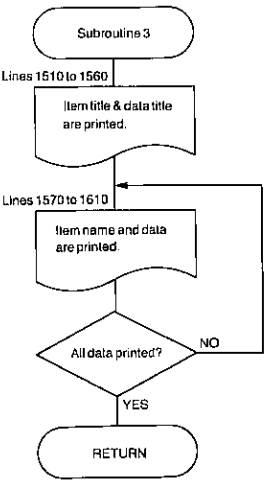
■ FLOWCHART


DEF A
Line 20
Program title is displayed.
Line 40
Subroutine 1
Lines 90 to 170
MJ=6, 12, or 18?
YES
Character size is determined.
Line 180
NO
Character size is determined.
Line 190
Graph title is printed.
Lines 230 to 250
Percentage is calculated for each item.
All data calculated?
NO
YES
Lines 260 to 290
Angle calculation is performed.
All data calculated?
NO
YES
Lines 300 to 310
Starting x, y coordinates are calculated for internal & external circles.
2
Lines 350 to 360
Angles of sector in x, y directions are calculated.
1
1
Lines 390 to 410
Cutout sections in x, y directions are calculated.
Lines 450 to 610
Circles are plotted.
Lines 630 to 640
MJ=6 or 12?
YES
Character size is determined.
NO
Line 680
Subroutine 2
Lines 690 to 860
Coordinates are calculated.
Lines 870 to 880
Item name and component ratio are printed.
All data completed?
NO
2
YES
Line 930
Tabulation required?
YES
Subroutine 3
NO
END


Subroutine 1
Lines 970 to 990
Graph title & No. of data are input.
Line 990
BASIC arrays are secured.
Lines 1000 to 1010
Item title & data title are input.
Lines 1030 to 1060
Item names and data are input.
Data input completed?
NO
YES
Line 1080
Need for data correction is determined.
Lines 1110 to 1140
Correction required?
NO
YES
Data No. of data to be corrected & correct data are input.
3


3
Line 1150
Need for data sorting is determined.
Data sorting required?
NO
YES
Lines 1180 to 1360
Data sorting is performed.
Line 1370
Need for doughnut-shaped graph is determined.
Line 1410
Need for cutout section is determined.
Cutout required?
NO
YES
Line 1440
Data No. of item requiring cutout is input.
Line 1460
Need for tabulation is determined.
RETURN


Subroutine 2
Lines 1630 to 1760
Coordinates are calculated
PT=2, 3, 4, 7, or 9?
YES
Lines 1800 to 1840
Item name & composition ratio are printed.
RETURN
NO
Lines 1850 to 1910
Coordinates are calculated.
Lines 1920 to 1960
Item name & composition ratio are printed.
RETURN


Subroutine 3
Lines 1510 to 1560
Item title & data title are printed.
Lines 1570 to 1610
Item name and data are printed.
All data printed?
NO
YES
RETURN

PROGRAM TITLE: Transfer of Program File

Required Peripheral Equipment
CE-130T

In the past, only cassette (micro cassette) tape was available as an external memory for pocket computers. But now a floppy disk in a personal computer can be used as an external memory for the PC-1460 pocket computer through the medium of the CE-130T level converter.
Here, transfer of program files to and from the PC-1460 pocket computer is explained using the PC-7000 as an example of the personal computers.

■ PREPARATION

- In addition to the PC-7000 personal computer, you must prepare a cable for connecting the CE-130T to the RS-232C interface of the PC-7000. The following table shows the cable requirements and pin connections necessary for interfacing.

CE-130T

Signal name	Pin No.
—	1
SD	2
RD	3
RS	4
CS	5
DR	6
SG	7
CD	8
RR	11
ER	20

Male connector
DB-25P or equivalent

PC-7000

Pin No.	Signal name
1	FG
2	SD
3	RD
4	RS
5	CS
6	DR
7	SG
8	CD
20	ER

Female connector
DB-25S or equivalent

..... denotes that the connection between these two pins is not mandatory.

- UPLOADING PROGRAM FILE (PC-1460 → PC-7000) First, the method of transferring a program file from the PC-1460 to the PC-7000 is introduced here. The PC-7000 will in turn store the data received from the PC-1460 in a 5-¼-inch floppy disk. The contents of the program file being received can be confirmed on the screen of the PC-7000.

■ OPERATING PROCEDURE

PC-1460

Program input

Input a program to be transferred to the PC-7000.

PC-7000

Program input

Input Program 1-1. (See below.)

Preparation for Program File Transfer

Set PRO or RUN mode.
CLOSE [ENTER]
OPEN "1200, N, 8, 1, A, L, &1A" [ENTER]
→ SAVE command is now executable.

⇨

Program File Receiving

RUN [↵]
→ The computer executes Program 1-1.
→ The computer waits for receipt of program file.

⇦

Program File Transfer

SAVE [ENTER].
→ The computer transfers program file.

⇨

The computer receives program file.
→ The contents of the received file are displayed one line at a time on the screen.
Input file name of received program.
→ Storage of program file in floppy disk is now
completed.

■ PROGRAM LIST 1-1

```
10 CLEAR:DIM A$(1000)
20 CLOSE:OPEN "COM1:1200,N,8,1" AS #1
30 Z$=INPUT$(1,1):IF Z$=CHR$(&H1A) THEN 100
40 LINE INPUT #1,A$
50 A$=Z$+A$
60 PRINT A$
70 A$(I)=A$
80 I=I+1
90 GOTO 30
100 INPUT "FILE NAME=";FI$
110 OPEN "O",#2,FI$
120 FOR J=0 TO I-1
130 PRINT #2,A$(J)
140 NEXT J
150 CLOSE
160 END
```

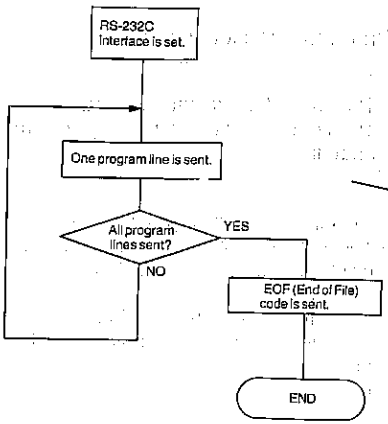
No programming for uploading is required on the part of the PC-1460.

■ MEMORY CONTENTS

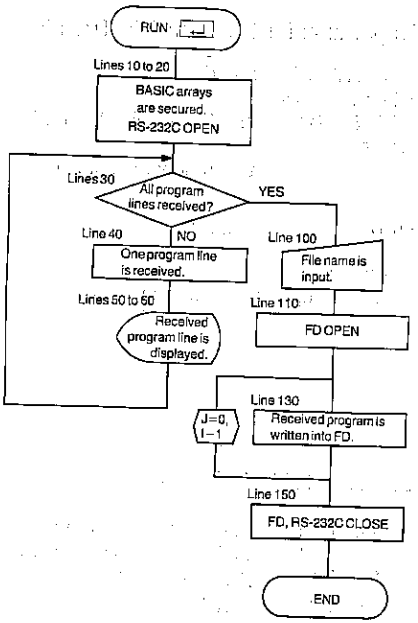
I	Counter
J	Loop counter
A$	For input of one program line
Z$	√
FI$	File name
A$(1000)	√

■ FLOWCHARTS

PC-1460 (Manual setting)



PC-7000



- DOWNLOADING PROGRAM FILE (PC-7000 → PC-1460)
  The method of transferring the program file in the floppy disk of the PC-7000 to the PC-1460 is covered here. The contents of the program file being transferred can be confirmed on the screen of the PC-7000.

■ OPERATING PROCEDURE

PC-7000

Program input

Input Program 2-1. (See below.)

Program File Transfer

RUN ↵
→ The computer executes Program 2-1. Input file name of program to be transferred.
→ The contents of program file being transferred are displayed one line at a time.
→ The computer transfers program file.

PC-1460

Preparation for Receiving Program File

Set PRO or RUN mode.
CLOSE [ENTER]
OPEN "1200, N, 8, 1, A, L, &1A" [ENTER]
→ LOAD command is now executable.

Program File Receiving

LOAD [ENTER]
→ The computer waits for receipt of program file.
The computer receives program file.
→ End of transfer

■ PROGRAM LIST 2-1

```
10 CLOSE:OPEN "COM1:1200,N,8,1" AS #1
20 INPUT "FILE NAME=";FI$
30 OPEN "I",#2,FI$
40 IF EOF(2)=-1 THEN PRINT #1,CHR$(&H1A);:CLOSE:END
50 LINE INPUT #2,A$
60 PRINT A$
70 PRINT #1,A$+CHR$(&HD);
80 GOTO 40
```

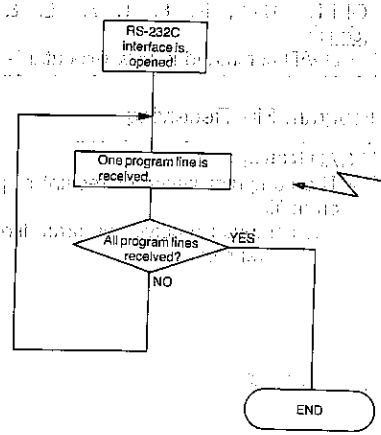
No programming for downloading is required on the part of the PC-1460.

■ MEMORY CONTENTS

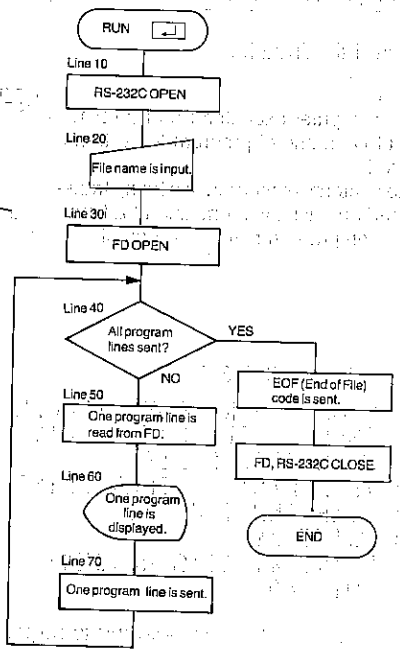
A$	For output of one program line
FI$	File name

■ FLOWCHARTS

PC-1460 (Manual setting)



PC-7000



PROGRAM TITLE: Three-Dimensional Graph

Required Peripheral Equipment
CE-140P or CE-515P
(or CE-516P) and CE-516L

You probably have been working hard on letters and numbers with your computer. Why not take a break for a while? However, do not just absent-mindedly look up at the ceiling like many are prone to do. Your computer has many functions with which you can enjoy yourself even while taking a rest. How about drawing a picture for a change? It may be fun to create a figure of geometrical pattern, though it will be far from drawing a real picture. You can relax while remembering those days when you used to play at making patterns in primary school. This should be more enjoyable than just daydreaming.

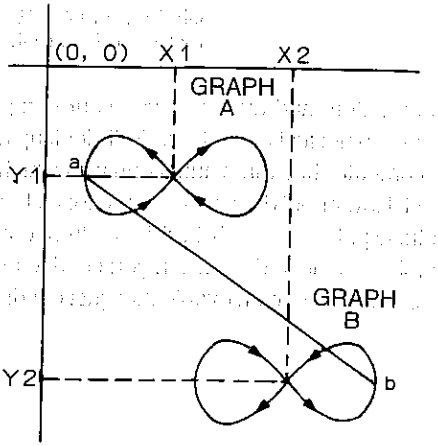
■ HOW TO OPERATE

1. Press DEF A . (Program starts)
2. Input one of the following four values (color numbers) of the color to be used according to the display.
   0: black 1: purple 2: green 3: red
3. Input each of the following four coordinates according to the display.
   (Y2≤−50, Y2≤0)
   X1, Y1 } Coordinates×and y at center of graphics
   X2, Y2 } Maximum coordinates×and y of graphics

   Note: X2 must be within 380, and Y2 within -999.

When the above input is completed, the graph is output on the printer. Then, the program ends.

■ REMARKS



■ EXAMPLE

Draw two graphs using the data shown in the following table.

No.	Color	X1	Y1	X2	Y2
1	Red	220	−150	100	−50
2	Purple	220	0	100	−100

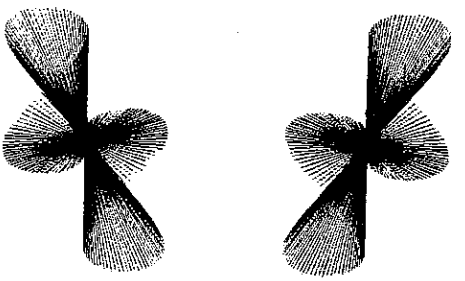
■ When using CE-515P (or CE-516P)

Modify line 10 of the program list on page 339 as shown below.

```
10:"A": CLOSE : OPEN "1200,N,8,1,A,L":
   CONSOLE 80
```

■ PRINTOUT

(Printed out in color. Refer to page I.)



■ KEY OPERATION SEQUENCE

<Example 1>

1. DEF A

```
COLOR (0-3)=_
```

2. 3 ENTER

```
X1=_
```

3. 220 ENTER

```
Y1=_
```

⋮

Input data in the same manner as above.

⋮

4. -50 ENTER

```
>
```

<Example 2>

1. DEF A

```
COLOR (0-3)=_
```

2. 1 ENTER

```
X1=_
```

3. 220 ENTER

```
Y1=_
```

⋮

Input data in the same manner as above.

⋮

4. -100 ENTER

```
>
```

■ PROGRAM LIST

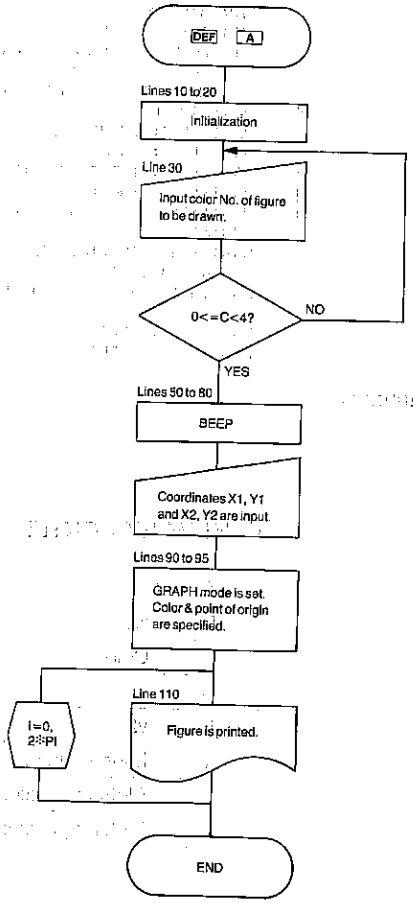
```
10:"A": CONSOLE 80
15:LTEXT : LPRINT
20:CLEAR : WAIT 0: RADIAN
30:BEEP 1: INPUT "COLOR (0-3)=";C:C=
   INT C: IF 0<=C AND C<4 THEN 50
40:GOTO 30
50:BEEP 1: INPUT "X1=";A
60:BEEP 1: INPUT "Y1=";D
70:BEEP 1: INPUT "X2=";B
80:BEEP 1: INPUT "Y2=";E
90:GRAPH : GLCURSOR (100,0)
95:SORGN
100:FOR I=0 TO 2*PI STEP .05:X=COS I*10
   0:Y=X*SIN I
110:LLINE (A+X,D-Y)-(B+X,E+Y),0,C: NEXT
   I
120:LTEXT : LPRINT : LF 12
130:COLOR 0: BEEP 2: END
```

249 Bytes

■ MEMORY CONTENTS

A	x1
B	x2
C	Color
D	y1
E	y2
I	Loop counter
X	Plotting position of x-axis
Y	Plotting position of y-axis

■ FLOWCHART


DEF A
Lines 10 to 20
Initialization
Line 30
Input color No. of figure to be drawn.
0<=C<4?
NO
YES
Lines 50 to 80
BEEP
Coordinates X1, Y1 and X2, Y2 are input.
Lines 90 to 95
GRAPH mode is set. Color & point of origin are specified.
Line 110
I=0, 2*PI
Figure is printed.
END

PROGRAM TITLE: Number Guessing Game

This program is designed to allow you to play a game of guessing a 3-digit number to be generated randomly from the computer. Don't study too much for examinations. Try this game for a change. Now, let us see how many attempts you must make before you can make a hit!

■ HOW TO OPERATE

1. Press DEF A . (Program starts)
2. "X=" will be displayed on the screen. Now, input a 3-digit number which you think the computer might have generated. Then, the screen will display the number of attempts you made and the 3-digit number you entered, followed by a comment (about 1 second later).

   For example,

   - Comment: 1 1

   If the display reads as shown above, the first number (1) following comment tells you that one digit of the 3-digit number you have input matches the random number generated with respect to its digit position and value. The second number (1) tells you that one digit of the 3-digit number you have input matches the random number generated with respect to its value only.

   - Comment: 3 0

   If the display reads as shown above, all three digits of the number you have input match the random number generated with respect to their digit positions and values. When you make a hit, the message "VERY GOOD!" and the number of attempts you made before the hit appear on the screen. Then, the program ends.

   Note: Remember you can only input a 3-digit number.

■ KEY OPERATION SEQUENCE

1. DEF A

```
X=_
```

2. 123 ENTER

```
 1          123
Comment:0     1
X=_
```

3. 145 ENTER

```
 2          145
Comment:1     0
X=_
```

⋮

Input data in the same manner as above.

⋮

4. 305 ENTER

```
 6          305
Comment:3     0
VERY GOOD !   6
>
```

■ PROGRAM LIST

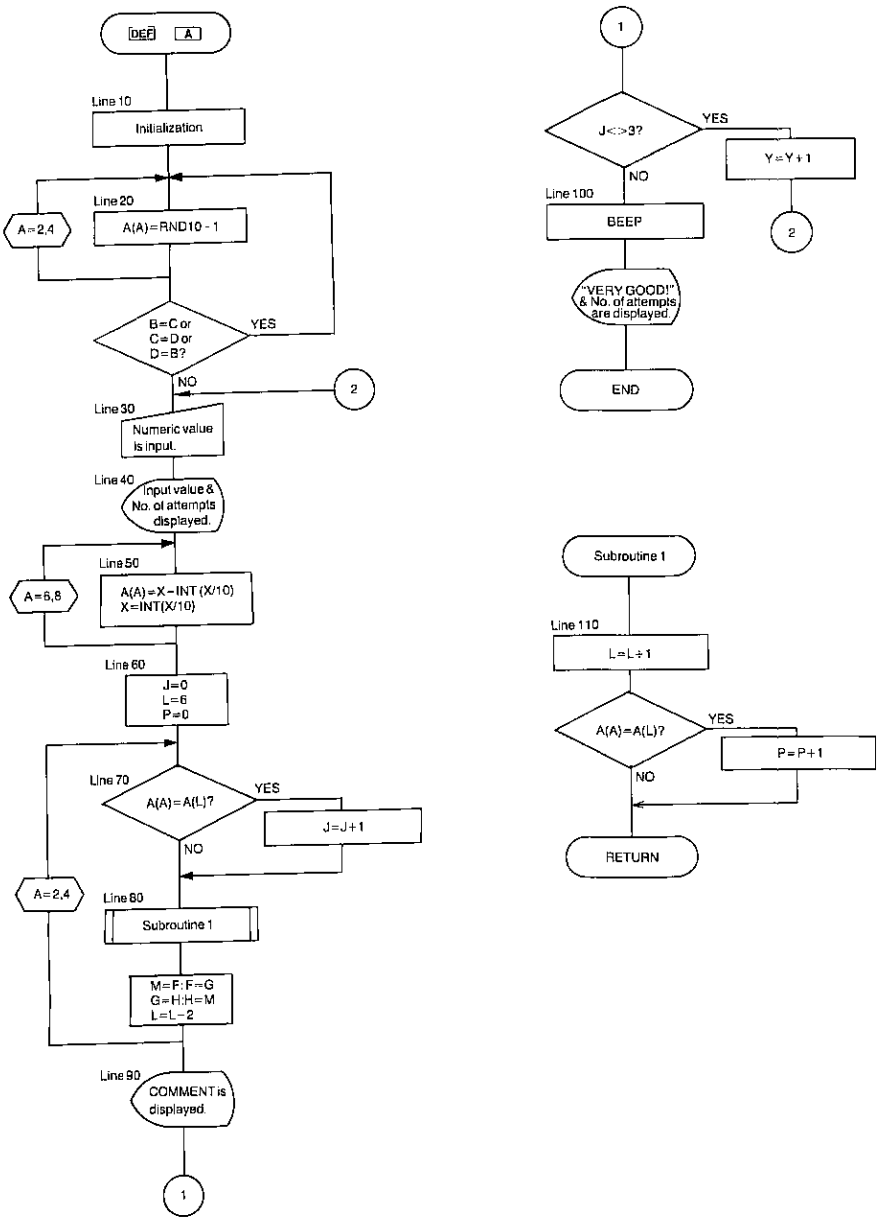
```
10:"A":CLEAR :RANDOM :Y
   =1
20:FOR A=2 TO 4:A(A)=
   RND 10-1:NEXT A:IF (
   B=C)+(C=D)+(D=B)<>0
   THEN 20
30:BEEP 1:INPUT "X=";X
40:USING :PAUSE Y,X
50:FOR A=6 TO 8:A(A)=X-
   INT (X/10)*10:X= INT
   (X/10):NEXT A
60:J=0:L=6:P=0
70:FOR A=2 TO 4:IF A(A)
   =A(L) LET J=J+1
80:GOSUB 110:GOSUB 110:
   M=F:F=G:G=H:H=M:L=L-
   2:NEXT A
90:PAUSE "Comment:";
   USING "####";J;P:IF
   J<>3 LET Y=Y+1:GOTO
   30
100:BEEP 2:PRINT "VERY G
   OOD ! ";Y:END
110:L=L+1:IF A(A)=A(L)
   LET P=P+1
120:RETURN
```

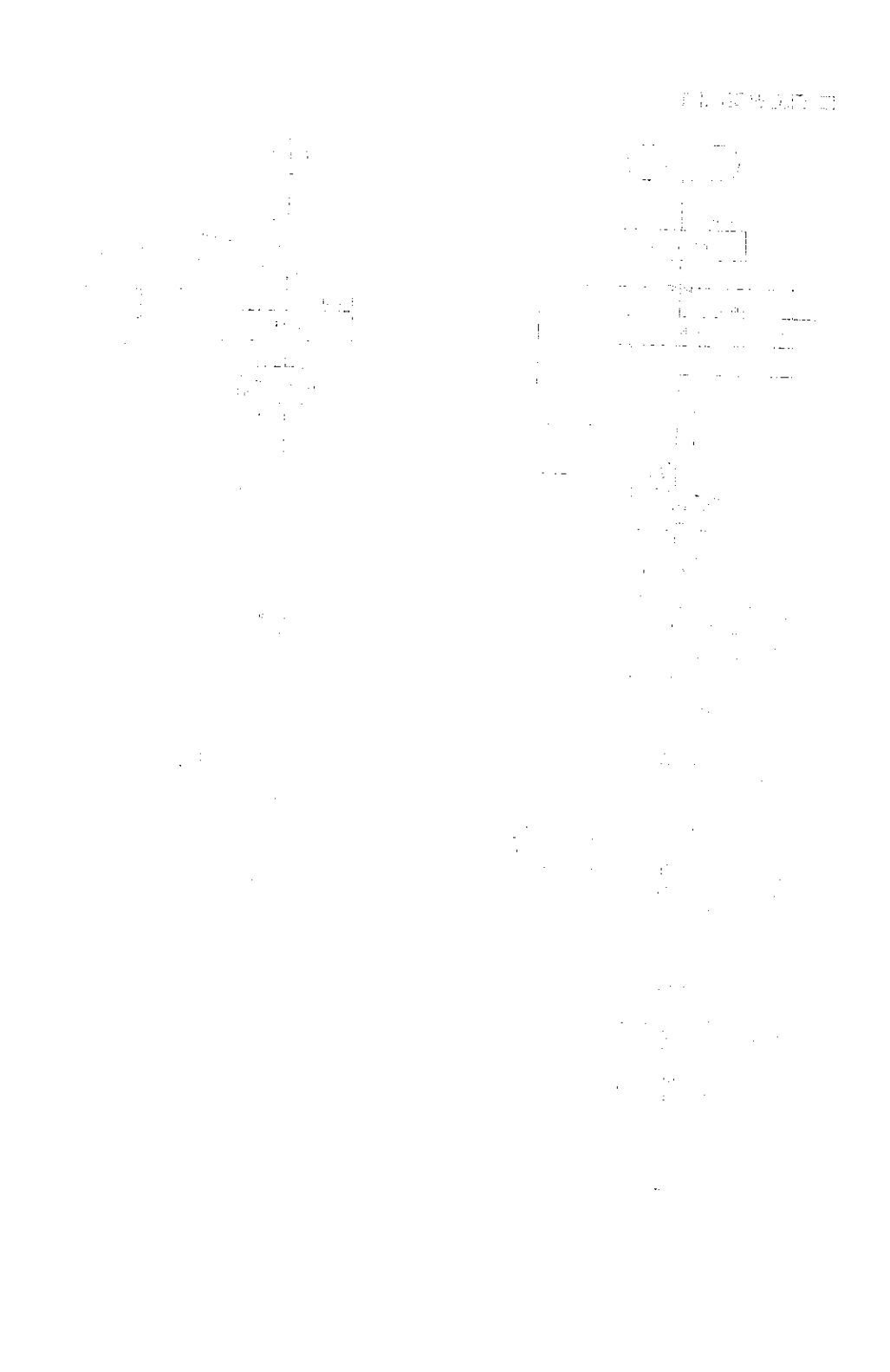
309 Bytes

■ MEMORY CONTENTS

A	√
B	√
C	3-digit number
D	√
F	√
G	√
H	√
J	Comment
L	√
M	√
P	Comment
X	Input value
Y	Number of attempts

■ FLOWCHART


DEF A
Line 10
Initialization
Line 20
A=2,4
A(A)=RND10 - 1
B=C or C=D or D=B?
YES
NO
2
Line 30
Numeric value is input.
Line 40
Input value & No. of attempts displayed.
Line 50
A=6,8
A(A)=X-INT(X/10)
X=INT(X/10)
Line 60
J=0
L=6
P=0
Line 70
A(A)=A(L)?
YES
J=J+1
NO
A=2,4
Line 80
Subroutine 1
M=F:F=G
G=H:H=M
L=L-2
Line 90
COMMENT is displayed.
1
1
J<>3?
YES
Y=Y+1
NO
Line 100
BEEP
2
"VERY GOOD!" & No. of attempts are displayed.
END
Subroutine 1
Line 110
L=L+1
A(A)=A(L)?
YES
P=P+1
NO
RETURN



INDEX

& 79
* 88
+ 88
− 88
/ 88
^ 88
√ 221
< 88
◀ 59
<= 88
<> 88
= 88
> 88
▶ 59
>= 88
π 216
↑ 96
↓ 96
A( ) variables 86
ABS 217
ACS 217
AHC 217
AHS 217
AHT 217
ALL RESET 10
AND 89
AREAD 146
ASC 223
ASN 218
ATN 218
Array variables 83
Auto off (Auto Power Off) 17
BASIC 241
BASIC key 7
BASIC mode 8
BEEP 148
CA key 263
CAL key 7
CAL mode 8
C.CE key 263
CE-126P 105, 169
CE-140P 113, 171
CE-515P 113
Cell replacement 11
CHAIN 149
CIRCLE 195
CHR$ 223
CLEAR 151
CLOAD 129
CLOAD? 130
CLOSE#1 225
Clear key 19
COLOR 198
CONSOLE 226
CONT 131
COS 218
CROTATE 200
CSAVE 132
CSIZE 201
CUR 218
Cursor 9
Cassette tape 112
Character Code Chart 255
Commands 94, 126, 129
DATA 152
DEF key 103
DEG 218
DEGREE 153
DELETE 133
DELete key 97
DIM 154
Direct calculation feature 76
DMS 218
Debugging 247
Display 22
END 156
ENTER key 58
EXP 219

Edition calculations 58
Editing programs 96
Error Messages 251
Expressions 87
FACT 219
Fixed variables 81
FOR..TO...STEP 157
Formatting output 257
Functions 92
GLCURSOR 202
GOSUB 159
GOTO 134, 160
GRAD 161
GRAPH 203
Hard cover 4
HCS 219
HSN 219
HTN 219
IF...THEN 162
INKEY$ 214
INPUT 163
INPUT# 165
INPUT#1 227
INSert key 61
INT 219
LEFT$ 223
LEN 223
LET 168
LF 205
LLINE 206
LIST 135
LLIST 136, 228
LN 219
LOAD 230
LOG 219
LPRINT 169, 171, 232
LTEXT 209
Labeled programs 103
Last answer feature 70
Limits of numbers 69
Line numbers 93
Linear regression 39
Logical expressions 89
MATRIX operations 46
MEM 215
MID$ 223
Maintenance 249
Manual calculations 57
Masks 257
MDF 173
Memory Protection 142
MERGE 138
NEW 141
NEXT 174
NOT 89
Numeric expressions 88
Numeric functions 217
Numeric variables 81
ON (Start up) 17
ON...GOSUB 175
ON...GOTO 176
OPEN 235
OPEN$ 237
OR 89
Operator priority (BASIC) mode 261
Operators 88
P ↔ NP 106
PAINT 210
PASS 142
PAUSE 177
PI 216
PRINT 179
PRINT# 181
PRINT#1 238
PROgram mode 7
Parentheses 91
POL 220
Printer 105
Priority (CAL mode) 31
Program 93
Pseudovariables 214
RADIAN 183
RAM Card 117
RAM Card Battery replacement 14
RANDOM 184
RCP 220

READ 185
REC 220
Relational expressions 88
REM 186
RENUM 143
RESET 10
RESTORE 187
RETURN 188
RIGHT$ 224
RLINE 212
RND 220
RUN 145
RUN mode 92
Range of numbers 69
Relational expressions 88
ROT 221
SAVE 240
Scientific notation 68
Serial I/O function 225
SGN 221
SHIFT key 18
Simple variable 82
SIN 211
Single-variable statistics 38
SQR 221
SORGN 213
SQU 221
STOP 189
STR$ 224
Statements 93
Statistical calculations 37
String expressions 88
String function 223
String variables 81
Subroutines 159
TAN 222
TEN 222
TEXT 242
TROFF 190
TRON 191
Tape Recorder 107
Template 104
Troubleshooting 245
Two-variable statistics 39
USING 192
VAL 224
Variables 80
Verbs 94, 127, 146
WAIT 194

SHARP CORPORATION

OSAKA, JAPAN

1986 © SHARP CORPORATIO

PRINTED IN JAPAN/IMPRIMÉ AU JAP
7F0.2T(TINSE1057ECZZ)